LAPORAN PRAKTIKUM
ALGORITMA PEMROGRAMAN
MODUL 12 & 13
PENGURUTAN DATA

Oleh :

Dimas Bagus Firmansyah

2311102002

S1 IF 11 02

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

12.3 Ide Algoritma Selection Sort

Pengurutan secara seleksi ini idenya adalah mencari nilai ekstrim pada sekumpulan data, kemudian meletakkannya 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.

12.4 Ide Algoritmma 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.

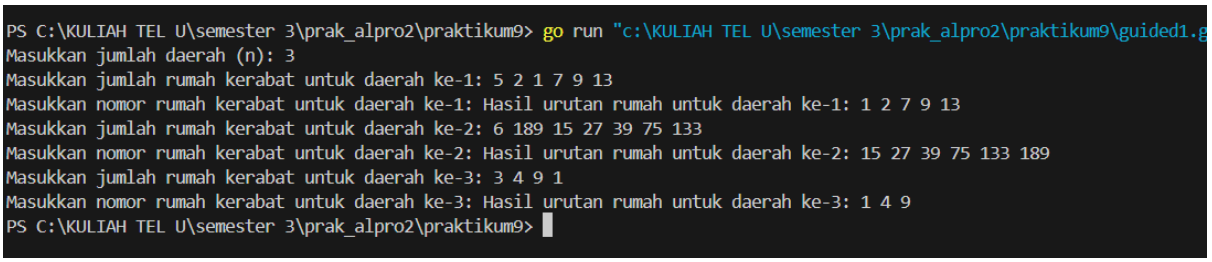
II. GUIDED

GUIDED 1
SOURCE CODE

```
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()
        }
    }
```

SCREENSHOOT PROGRAM



DESKRIPSI PROGRAM

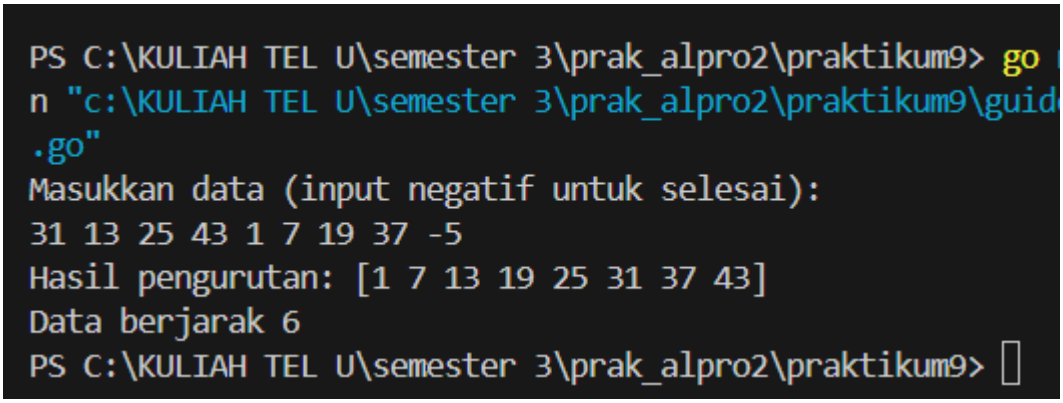
Pengguna diminta untuk memasukkan jumlah wilayah terlebih dahulu, kemudian jumlah rumah di setiap wilayah, dan kemudian jumlah rumah di setiap wilayah. Program memeriksa apakah masukan berada dalam rentang yang sesuai (jumlah lingkungan dari 1 hingga 999, jumlah rumah dari 1 hingga 999,999). Setelah semua data dimasukkan, program menggunakan algoritma pengurutan selektif untuk mengurutkan alamat jalan di setiap area dan mengeluarkan hasilnya dalam urutan yang diurutkan. Program ini juga menangani validasi masukan untuk memastikan bahwa data yang dimasukkan memenuhi spesifikasi.

GUIDED 2

```
package main

import (
    "fmt"
    "math"
)

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 isDataConsistentlySpaced(arr []int) (bool, int) {
    if len(arr) < 2 {
        return true, 0
    }

    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
        }
    }

    return true, diff
}

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

    fmt.Println("Masukkan data (input negatif untuk selesai):")
    for {
        fmt.Scan(&input)
        if input < 0 {
            break
        }
        data = append(data, input)
    }

    insertionSort(data)

    isConsistent, diff := isDataConsistentlySpaced(data)

    fmt.Println("Hasil pengurutan:", data)
    if isConsistent {
        fmt.Printf("Data berjarak %d\n", diff)
    } else {
        fmt.Println("Data berjarak tidak tetap")
    }
}
```

SCREENSHOOT PROGRAM



DESKRIPSI PROGRAM

Program pada atas merupakan pelaksanaan berbasis Go yg membaca sekumpulan sapta integer menurut pengguna, mengurutkan sapta tadi memakai algoritma **insertion sort**, & lalu menilik apakah selisih antar elemen pada array yg terurut tadi merupakan tetap (konsisten). Pengguna diminta buat memasukkan sapta satu per satu, menggunakan input negatif dipakai menjadi frekuwensi buat menghentikan masukan. Setelah sapta diurutkan, acara menilik apakah selisih mutlak antara elemen bertetangga pada array merupakan sama. apabila selisihnya tetap, acara mencetak nilai selisih tadi; apabila nir, acara memberi memahami bahwa data nir mempunyai jeda yg tetap. Hasil pengurutan & analisis jeda ditampilkan pada pengguna.

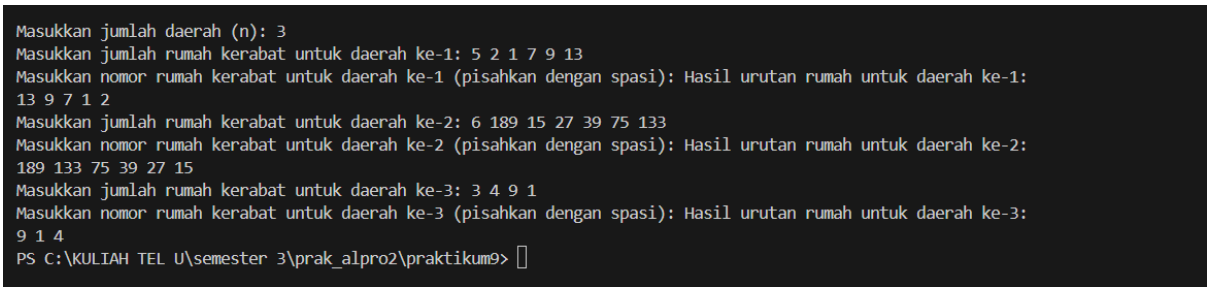
III. UNGUIDED

UNGUIDED 1

```
package main

import (
   "fmt"
   "sort"
)

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 (pisahkan
dengan spasi): ", i+1)
      for j := 0; j < m; j++ {
         fmt.Scan(&houses[j])
      }

      // Pisahkan ganjil dan genap
      var odd, even []int
      for _, house := range houses {
         if house%2 == 0 {
            even = append(even, house)
         } else {
            odd = append(odd, house)
         }
      }

      // Urutkan ganjil secara menurun dan genap secara menaik
      sort.Sort(sort.Reverse(sort.IntSlice(odd))) // Ganjil menurun
      sort.Ints(even)                             // Genap menaik
```

```
        // Cetak hasil sesuai format
        fmt.Printf("Hasil urutan rumah untuk daerah ke-%d:\n", i+1)
        for _, house := range odd {
            fmt.Printf("%d ", house)
        }
        for _, house := range even {
            fmt.Printf("%d ", house)
        }
        fmt.Println()
    }
}
```

SCHREENSHOOT PROGRAM



DESKRIPSI PROGRAM

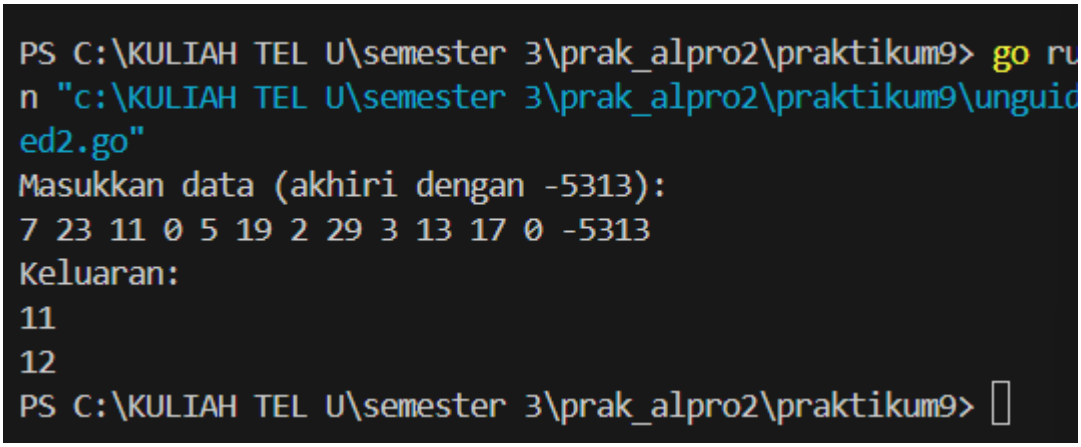
Pengguna diminta **buat** memasukkan jumlah **wilayah** (n), **lalu buat** setiap **wilayah**, mereka memasukkan jumlah **tempat tinggal** (m) **& angka** rumahnya. Program ini memisahkan **angka tempat tinggal gasal &** genap, **kemudian** mengurutkan **angka gasal** secara menurun **& angka** genap secara menaik. Hasilnya dicetak **pada** format **pada** mana **angka gasal** dicantumkan lebih dulu, diikuti **sang angka** genap. Program **jua** memvalidasi masukan **supaya** jumlah **wilayah & tempat tinggal** memenuhi rentang **yg sudah** ditentukan, yaitu n **pada** antara 1 **sampai** 999 **&** m **pada** antara 1 **sampai** 999,999.

GUIDED 2

```
package main

import (
    "bufio"
    "fmt"
    "os"
    "sort"
    "strconv"
    "strings"
)

func main() {
    scanner := bufio.NewScanner(os.Stdin)
```

```
    // Baca jumlah data
    fmt.Println("Masukkan data (akhiri dengan -5313):")
    scanner.Scan()
    input := scanner.Text()

    // Parsing input menjadi slice
    data := strings.Fields(input)
    numbers := []int{}

    for _, v := range data {
        num, err := strconv.Atoi(v)
        if err != nil {
            fmt.Println("Input tidak valid:", v)
            return
        }
        if num == -5313 {
            break
        }
        numbers = append(numbers, num)
    }

    // Proses data
    result := []int{}
    currentNumbers := []int{}

    for _, num := range numbers {
        if num == 0 {
            // Urutkan data yang sudah dibaca
            sort.Ints(currentNumbers)

            // Hitung median
            length := len(currentNumbers)
            if length%2 == 1 {
                // Jika jumlah data ganjil
                median := currentNumbers[length/2]
                result = append(result, median)
            } else {
                // Jika jumlah data genap
                median := (currentNumbers[length/2-1] + currentNumbers[length/2])
/ 2
                result = append(result, median)
            }
        } else {
            // Tambahkan data baru
            currentNumbers = append(currentNumbers, num)
        }
    }

    // Cetak hasil
    fmt.Println("Keluaran:")
    for _, res := range result {
        fmt.Println(res)
    }
}
```

SCREENSHOOT PROGRAM



DESKRIPSI PROGRAM

Program ini membaca sekumpulan sapta menurut input pengguna sampai menemukan penanda akhir berupa '-5313'. Setiap kali angka '0' ada pada data, acara menghitung median menurut sapta yg sudah dimasukkan semenjak angka '0' terakhir atau semenjak awal. Median dihitung menggunakan mengurutkan sapta, lalu merogoh nilai tengah (bila jumlah sapta ganjil) atau homogen-homogen 2 nilai tengah (bila jumlah sapta genap). Hasil median menurut setiap segmen data dicatat & ditampilkan waktu proses selesai.

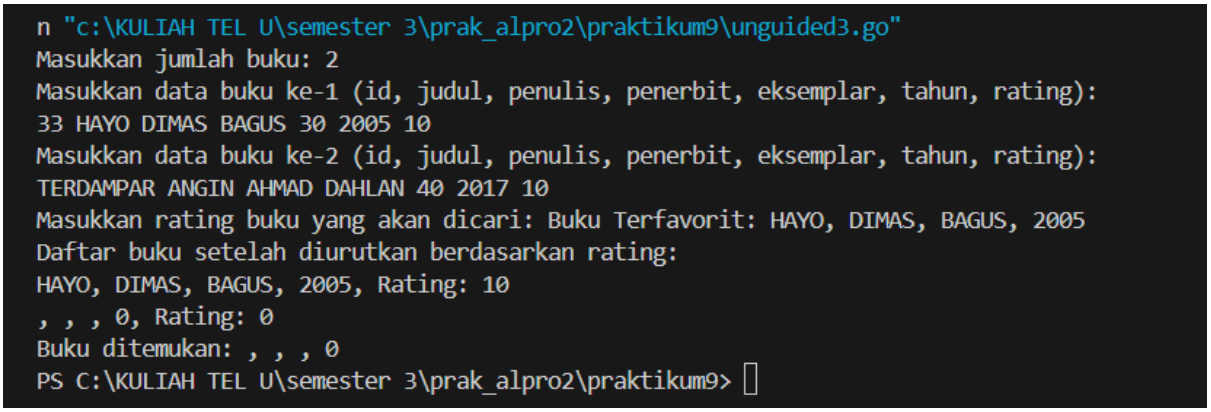
UNGUIDED 3

```
package main

import (
    "fmt"
    "sort"
)

const nMax = 7919

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

type DaftarBuku []Buku

// Subprogram untuk mencetak buku favorit
func CetakTerFavorit(pustaka DaftarBuku) {
    if len(pustaka) == 0 {
        fmt.Println("Tidak ada data buku.")
        return
    }
```

```
    // Cari buku dengan rating tertinggi
    terfavorit := pustaka[0]
    for _, buku := range pustaka {
        if buku.rating > terfavorit.rating {
            terfavorit = buku
        }
    }

    // Cetak buku favorit
        fmt.Printf("Buku Terfavorit: %s, %s, %s, %d\n", terfavorit.judul,
terfavorit.penulis, terfavorit.penerbit, terfavorit.tahun)
}

// Subprogram untuk mengurutkan buku berdasarkan rating secara menurun
func UrutBuku(pustaka *DaftarBuku) {
    sort.Slice(*pustaka, func(i, j int) bool {
        return (*pustaka)[i].rating > (*pustaka)[j].rating
    })
}

// Subprogram untuk mencari buku berdasarkan rating
func CariBuku(pustaka DaftarBuku, r int) {
    for _, buku := range pustaka {
        if buku.rating == r {
                fmt.Printf("Buku ditemukan: %s, %s, %s, %d\n", buku.judul,
buku.penulis, buku.penerbit, buku.tahun)
            return
        }
    }
    fmt.Println("Buku dengan rating yang diminta tidak ditemukan.")
}

func main() {
    var n, ratingCari int

    // Input jumlah buku
    fmt.Print("Masukkan jumlah buku: ")
    fmt.Scan(&n)

    if n <= 0 || n > nMax {
        fmt.Printf("Jumlah buku harus antara 1 hingga %d.\n", nMax)
        return
    }

    // Input data buku
    pustaka := make(DaftarBuku, n)
    for i := 0; i < n; i++ {
            fmt.Printf("Masukkan data buku ke-%d (id, judul, penulis, penerbit,
eksemplar, tahun, rating):\n", 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)
    }
```

```
    // Input rating yang akan dicari
    fmt.Print("Masukkan rating buku yang akan dicari: ")
    fmt.Scan(&ratingCari)

    // Cetak buku terfavorit
    CetakTerFavorit(pustaka)

    // Urutkan buku berdasarkan rating
    UrutBuku(&pustaka)

    // Cetak daftar buku setelah diurutkan
    fmt.Println("Daftar buku setelah diurutkan berdasarkan rating:")
    for _, buku := range pustaka {
        fmt.Printf("%s, %s, %s, %d, Rating: %d\n", buku.judul, buku.penulis,
buku.penerbit, buku.tahun, buku.rating)
    }

    // Cari buku berdasarkan rating
    CariBuku(pustaka, ratingCari)
}
```

SCREENSHOOT PROGRAM



DESKRIPSI PROGRAM

Program ini mengelola data **kitab pada** sebuah perpustakaan **menggunakan** fitur **buat** mencetak **kitab menggunakan** rating tertinggi, mengurutkan **kitab menurut** rating secara menurun, **&** mencari **kitab menurut** rating **eksklusif**. Data **kitab** dimasukkan **sang** pengguna **menggunakan** atribut **misalnya** ID, judul, penulis, penerbit, jumlah eksemplar, tahun, **&** rating. Setelah **seluruh** data diinput, **acara** menampilkan **kitab** terfavorit, mengurutkan **kitab menurut** rating, **kemudian** mencetak daftar **kitab yg sudah** diurutkan. Program **pula** memungkinkan pengguna mencari **kitab menurut** rating **eksklusif & menaruh output** pencarian.