LAPORAN PRAKTIKUM
ALGORITMA PEMROGRAMAN 2

MODUL 11

PENGURUTAN DATA

Oleh:

Ben Waiz Pintus Widyosaputro

2311102169

IF-11-02

S1 TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2024

I. DASAR TEORI

Selection Sort

Pengertian:

Selection Sort adalah algoritma pengurutan yang bekerja dengan cara mencari elemen terkecil (atau terbesar, tergantung urutan) dari array yang belum terurut dan menukarnya dengan elemen pertama dari array tersebut. Proses ini diulang untuk elemen berikutnya hingga seluruh array terurut.

Cara kerja:

1. Mulai dari elemen pertama, cari elemen terkecil di antara elemen yang belum terurut.
2. Tukar elemen terkecil tersebut dengan elemen pertama.
3. Pindah ke elemen berikutnya dan ulangi proses hingga semua elemen terurut.

Insertion Sort

Pengertian:

Insertion Sort adalah algoritma pengurutan yang membangun subarray terurut satu per satu. Algoritma ini mengambil elemen dari bagian array yang belum diurutkan dan menyisipkannya ke posisi yang tepat dalam subarray yang sudah terurut.

Cara kerja:

1. Anggap elemen pertama sudah terurut.
2. Ambil elemen berikutnya (key) dari bagian yang belum diurutkan.
3. Bandingkan key dengan elemen-elemen di subarray terurut dari belakang ke depan.
4. Geser elemen-elemen yang lebih besar satu posisi ke kanan.
5. Sisipkan key pada posisi yang tepat.
6. Ulangi proses hingga seluruh array terurut.

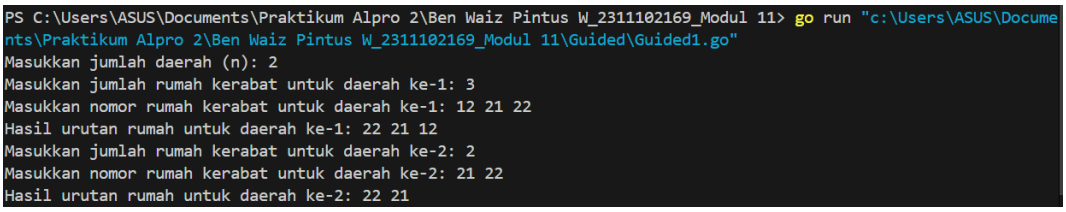
II. GUIDED

Guided 1

```
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 output



Penjelasan :

Program ini membaca sejumlah data nomor rumah kerabat di beberapa daerah, mengurutkannya, dan menampilkan hasilnya. Untuk setiap daerah, pengguna memasukkan jumlah rumah kerabat dan nomor rumah masing-masing. Program menggunakan algoritma **selection sort** untuk mengurutkan nomor rumah secara menurun (descending). Hasil pengurutan untuk setiap daerah ditampilkan dalam urutan terbesar ke terkecil.

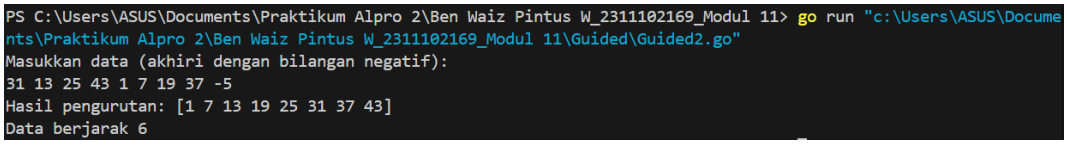
Guided 2

```
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")
    }
}
```

Output :



Penjelasan :

Program ini membaca sejumlah data bilangan bulat dari pengguna hingga bilangan negatif dimasukkan sebagai penanda akhir input. Data tersebut diurutkan secara menaik menggunakan algoritma **insertion sort**. Setelah data diurutkan, program memeriksa apakah selisih antara elemen-elemen dalam array memiliki jarak yang tetap. Jika data berjarak tetap, program menampilkan selisihnya jika tidak, program menyatakan bahwa data tidak berjarak tetap. Output program mencakup hasil pengurutan data serta informasi tentang konsistensi jarak antar elemen.

III. UNGUIDED

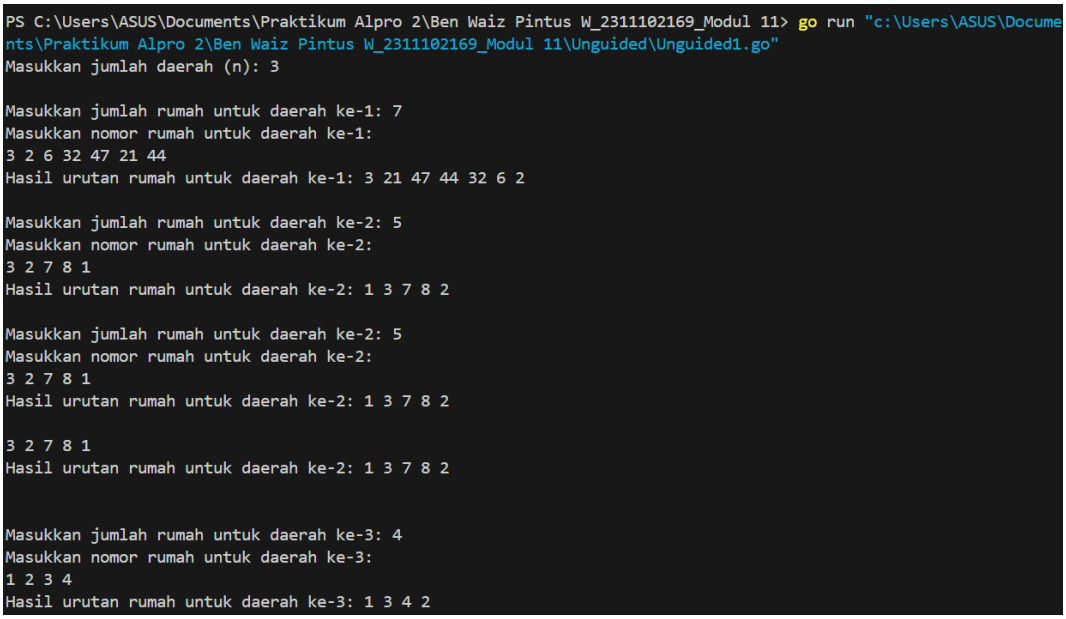
Unguided 1

```
//Ben Waiz Pintus W
//2311102169

package main

import (
    "fmt"
)

// Fungsi untuk mengurutkan array secara ascending menggunakan
selection sort
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] { // Cari elemen terkecil
                minIdx = j
            }
        }
        arr[i], arr[minIdx] = arr[minIdx], arr[i] // Tukar elemen
    }
}

// Fungsi untuk mengurutkan array secara descending menggunakan
selection sort
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] { // 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("\nMasukkan jumlah rumah untuk daerah ke-%d: ", i+1)
    fmt.Scan(&m)

    if m <= 0 {
        fmt.Println("Jumlah rumah harus lebih besar dari 0.")
        return
    }

    // Membaca array angka sebagai input
    fmt.Printf("Masukkan nomor rumah untuk daerah ke-%d:\n", i+1)
    houses := make([]int, m)
    for j := 0; j < m; j++ {
        fmt.Scan(&houses[j])
    }

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

    // Urutkan ganjil (ascending) dan genap (descending)
    selectionSortAsc(ganjil)
    selectionSortDesc(genap)

    // Cetak hasil
    fmt.Printf("Hasil urutan rumah untuk daerah ke-%d: ", i+1)
    for _, num := range ganjil {
        fmt.Printf("%d ", num)
    }
    for _, num := range genap {
        fmt.Printf("%d ", num)
    }
```

```
        fmt.Println()
    }
}
```

Screenshot output



Penjelasan

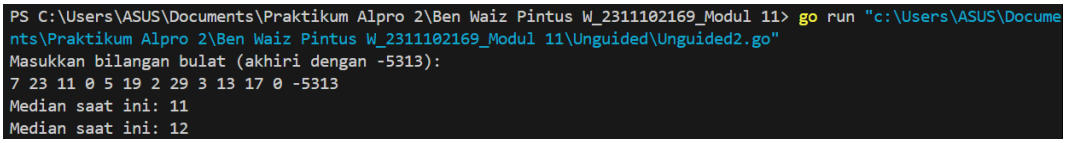
Program ini digunakan untuk mengurutkan nomor rumah kerabat di beberapa daerah berdasarkan bilangan ganjil dan genap. Pengguna diminta memasukkan jumlah daerah dan jumlah rumah di setiap daerah, diikuti oleh nomor rumah dalam bentuk array. Program memisahkan bilangan ganjil dan genap, kemudian mengurutkan bilangan ganjil secara menaik (ascending) dan bilangan genap secara menurun (descending) menggunakan algoritma **selection sort**. Hasil akhirnya menampilkan nomor rumah ganjil yang terurut diikuti oleh nomor rumah genap yang terurut untuk setiap daerah.

Unguided 2

```
//Ben Waiz Pintus W
//2311102169

package main

import (
    "fmt"
)

// Fungsi untuk mengurutkan array menggunakan insertion sort
func insertionSort(arr []int) {
    n := len(arr)
    for i := 1; i < n; i++ {
        key := arr[i]
        j := i - 1

        // Pindahkan elemen yang lebih besar dari key ke satu posisi di depan
        for j >= 0 && arr[j] > key {
            arr[j+1] = arr[j]
            j--
        }
        arr[j+1] = key
    }
}

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

func main() {
    var input int
    data := []int{} // Array untuk menyimpan data yang valid

    fmt.Println("Masukkan bilangan bulat (akhiri dengan -5313):")

    for {
        fmt.Scan(&input)
```

```
        if input == -5313 {
            // Marker untuk mengakhiri program
            break
        } else if input == 0 {
            // Jika menemukan angka 0, urutkan array dan cetak median
            insertionSort(data)
            if len(data) > 0 {
                median := getMedian(data)
                fmt.Println("Median saat ini:", median)
            }
        } else if input > 0 {
            // Hanya tambahkan bilangan bulat positif ke array
            data = append(data, input)
        }
    }
}
```

Screenshot output



Penjelasan

Program ini berfungsi untuk menerima input bilangan bulat dan menghitung median dari data yang sudah terurut. Pengguna memasukkan bilangan satu per satu, dan jika angka 0 dimasukkan, program akan mengurutkan data yang sudah ada menggunakan algoritma **insertion sort** dan menghitung median berdasarkan jumlah elemen yang ada (jika jumlahnya ganjil, median adalah elemen tengah, sedangkan jika genap, median adalah rata-rata dua elemen tengah). Program berlanjut hingga pengguna memasukkan angka -5313 sebagai penanda untuk menghentikan input. Data yang valid hanya yang berupa bilangan bulat positif, dan angka 0 berfungsi sebagai pemicu untuk menghitung dan menampilkan median saat itu.

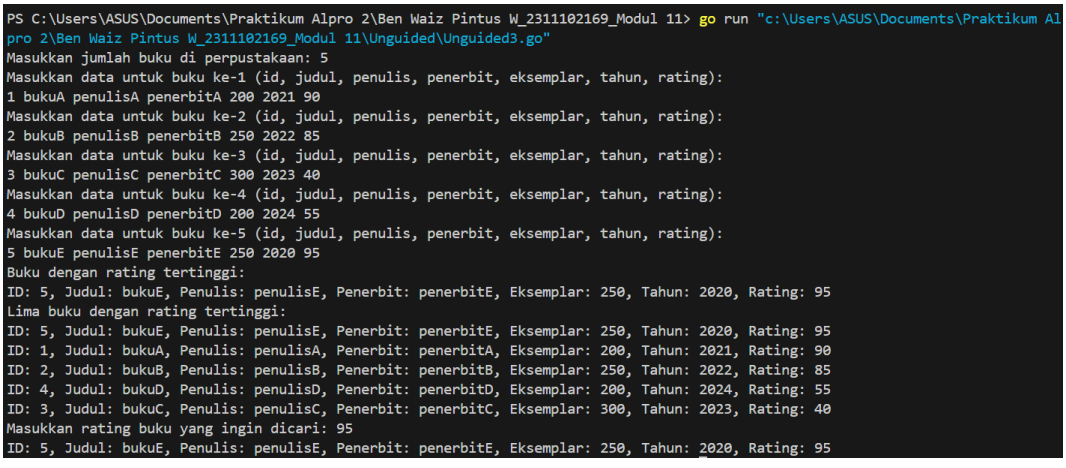
Unguided 3

```
package main

import (
    "fmt"
)

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

// Fungsi untuk menambahkan data buku ke pustaka
func DaftarkanBuku(pustaka *[]Buku, n int) {
    for i := 0; i < n; i++ {
        var buku Buku
         fmt.Printf("Masukkan data untuk buku ke-%d (id, judul, penulis, penerbit, eksemplar, tahun, rating):\n", i+1)
        fmt.Scan(&buku.id, &buku.judul, &buku.penulis, &buku.penerbit, &buku.eksemplar, &buku.tahun, &buku.rating)
        *pustaka = append(*pustaka, buku)
    }
}

// Fungsi untuk mencetak buku dengan rating tertinggi
func CetakFavorit(pustaka []Buku, n int) {
    if len(pustaka) == 0 {
        fmt.Println("Pustaka kosong.")
        return
    }
    terfavorit := pustaka[0]
    for _, buku := range pustaka {
        if buku.rating > terfavorit.rating {
            terfavorit = buku
        }
    }
    fmt.Println("Buku dengan rating tertinggi:")
    fmt.Printf("ID: %d, Judul: %s, Penulis: %s, Penerbit: %s, Eksemplar: %d, Tahun: %d, Rating: %d\n",
```

```
        terfavorit.id, terfavorit.judul, terfavorit.penulis, terfavorit.penerbit,
terfavorit.eksemplar, terfavorit.tahun, terfavorit.rating)
}

// Fungsi untuk mengurutkan array buku berdasarkan rating secara
descending
func UrutkanBuku(pustaka *[]Buku, n int) {
   for i := 1; i < len(*pustaka); 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
   }
}

// Fungsi untuk mencetak lima buku dengan rating tertinggi
func Cetak5Terbaik(pustaka []Buku, n int) {
   fmt.Println("Lima buku dengan rating tertinggi:")
   for i := 0; i < 5 && i < len(pustaka); i++ {
      buku := pustaka[i]
     fmt.Printf("ID: %d, Judul: %s, Penulis: %s, Penerbit: %s, Eksemplar:
%d, Tahun: %d, Rating: %d\n",
        buku.id, buku.judul, buku.penulis, buku.penerbit, buku.eksemplar,
buku.tahun, buku.rating)
   }
}

// Fungsi untuk mencari buku dengan rating tertentu
func CariBuku(pustaka []Buku, n int, r int) {
   ditemukan := false
   for _, buku := range pustaka {
      if buku.rating == r {
         ditemukan = true
            fmt.Printf("ID: %d, Judul: %s, Penulis: %s, Penerbit: %s,
Eksemplar: %d, Tahun: %d, Rating: %d\n",
                  buku.id, buku.judul, buku.penulis, buku.penerbit,
buku.eksemplar, buku.tahun, buku.rating)
      }
   }
   if !ditemukan {
      fmt.Println("Tidak ada buku dengan rating tersebut.")
   }
}
```

```
func main() {
    var n int
    fmt.Print("Masukkan jumlah buku di perpustakaan: ")
    fmt.Scan(&n)

    if n <= 0 || n > 7919 {
        fmt.Println("Jumlah buku harus antara 1 hingga 7919.")
        return
    }

    var pustaka []Buku

    // Input data buku
    DaftarkanBuku(&pustaka, n)

    // Cetak buku dengan rating tertinggi
    CetakFavorit(pustaka, n)

    // Urutkan buku berdasarkan rating
    UrutkanBuku(&pustaka, n)

    // Cetak lima buku dengan rating tertinggi
    Cetak5Terbaik(pustaka, n)

    // Cari buku dengan rating tertentu
    var rating int
    fmt.Print("Masukkan rating buku yang ingin dicari: ")
    fmt.Scan(&rating)
    CariBuku(pustaka, n, rating)
}
```

Screenshot output



Penjelasan

Program ini mengelola data buku dalam sebuah perpustakaan dengan menggunakan struktur data Buku. Program menerima input jumlah buku yang akan didaftarkan dan kemudian meminta pengguna untuk memasukkan informasi tentang setiap buku, seperti ID, judul, penulis, penerbit, jumlah eksemplar, tahun terbit, dan rating. Program menyediakan berbagai fitur, termasuk menampilkan buku dengan rating tertinggi, mengurutkan buku berdasarkan rating secara menurun, mencetak lima buku dengan rating tertinggi, dan mencari buku berdasarkan rating tertentu. Program juga membatasi jumlah buku yang dapat dimasukkan antara 1 hingga 7919, dan melakukan validasi input untuk memastikan bahwa data yang dimasukkan sesuai.