LAPORAN PRAKTIKUM
ALGORITMA PEMROGRAMAN 2

MODUL 12

PENGURUTAN DATA

Oleh:

ARNANDA SETYA NOSA PUTRA

2311102180

IF-11-02

S1 TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2024

I. DASAR TEORI

Selection Sort

Pengertian:

Selection Sort adalah algoritma pengurutan sederhana yang bekerja dengan cara memilih elemen terkecil (atau terbesar, tergantung urutan yang diinginkan) dari bagian array yang belum terurut, lalu menukarnya dengan elemen pertama dari bagian tersebut. Proses ini dilakukan secara berulang dengan berpindah ke elemen berikutnya, hingga seluruh elemen dalam array berada dalam urutan yang benar. Algoritma ini sangat intuitif dan sering digunakan dalam pembelajaran dasar struktur data karena konsepnya yang mudah dipahami.

Cara kerja:

1. Dimulai dari elemen pertama dalam array, algoritma mencari elemen terkecil di antara elemen-elemen yang belum diurutkan.
2. Setelah elemen terkecil ditemukan, algoritma menukarnya dengan elemen pertama dari bagian yang belum terurut.
3. Langkah ini diulangi dengan pindah ke elemen berikutnya, hingga seluruh array terurut.

Insertion Sort

Pengertian:

Insertion Sort adalah algoritma pengurutan yang bekerja dengan cara membangun subarray yang terurut secara bertahap. Algoritma ini mengambil elemen dari bagian array yang belum terurut, kemudian menyisipkannya ke posisi yang sesuai dalam subarray yang sudah terurut. Pendekatan ini menyerupai cara seseorang mengurutkan kartu dalam permainan, di mana setiap kartu baru disisipkan ke posisi yang tepat di antara kartu-kartu yang sudah diurutkan sebelumnya.

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.

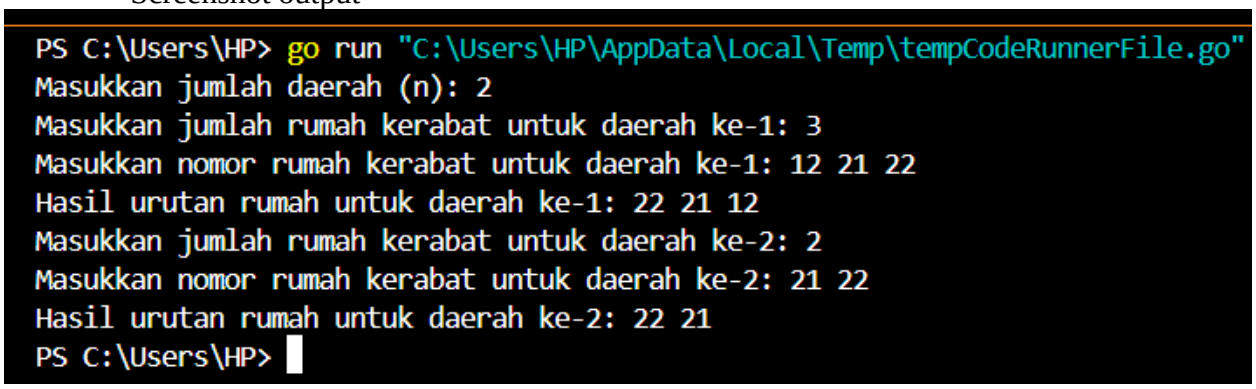
II. GUIDED

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 di atas untuk mengurutkan nomor rumah dalam beberapa daerah menggunakan algoritma selection sort. Program meminta pengguna memasukkan jumlah daerah (n) dan untuk setiap daerah, pengguna diminta memasukkan jumlah rumah kerabat (m) beserta nomor rumahnya. Nomor-nomor rumah tersebut diurutkan dalam urutan menurun dengan selection sort, dan hasilnya ditampilkan untuk setiap daerah. Program juga memiliki validasi untuk memastikan input n dan m berada dalam rentang yang diperbolehkan. Jika input tidak valid, program akan berhenti dan memberikan pesan kesalahan.

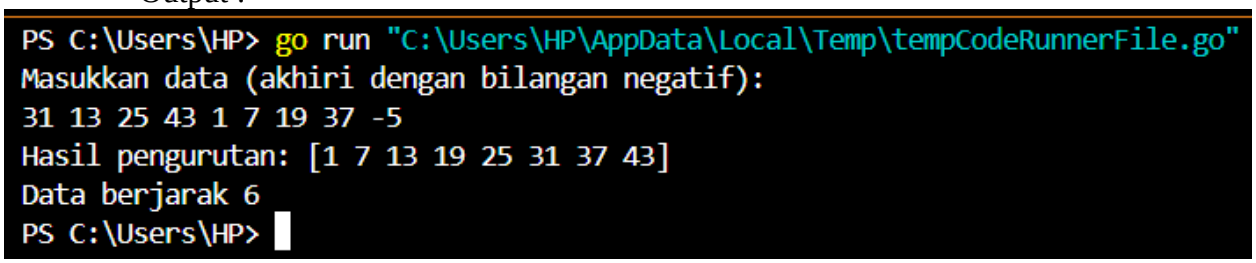
Guided 2

```
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 di atas menggabungkan fungsi *sorting* dengan *insertion sort* dan pemeriksaan apakah elemen-elemen dalam array memiliki jarak tetap (selisih konsisten). Program meminta pengguna memasukkan bilangan bulat secara bertahap hingga bilangan negatif dimasukkan sebagai tanda akhir input. Data yang dimasukkan diurutkan menggunakan algoritma *insertion sort*. Setelah itu, program memeriksa apakah selisih antar elemen yang sudah diurutkan konstan. Jika ya, program mencetak bahwa data berjarak tetap beserta nilai jaraknya. Jika tidak, program menyatakan data tidak berjarak tetap.

III. UNGUIDED

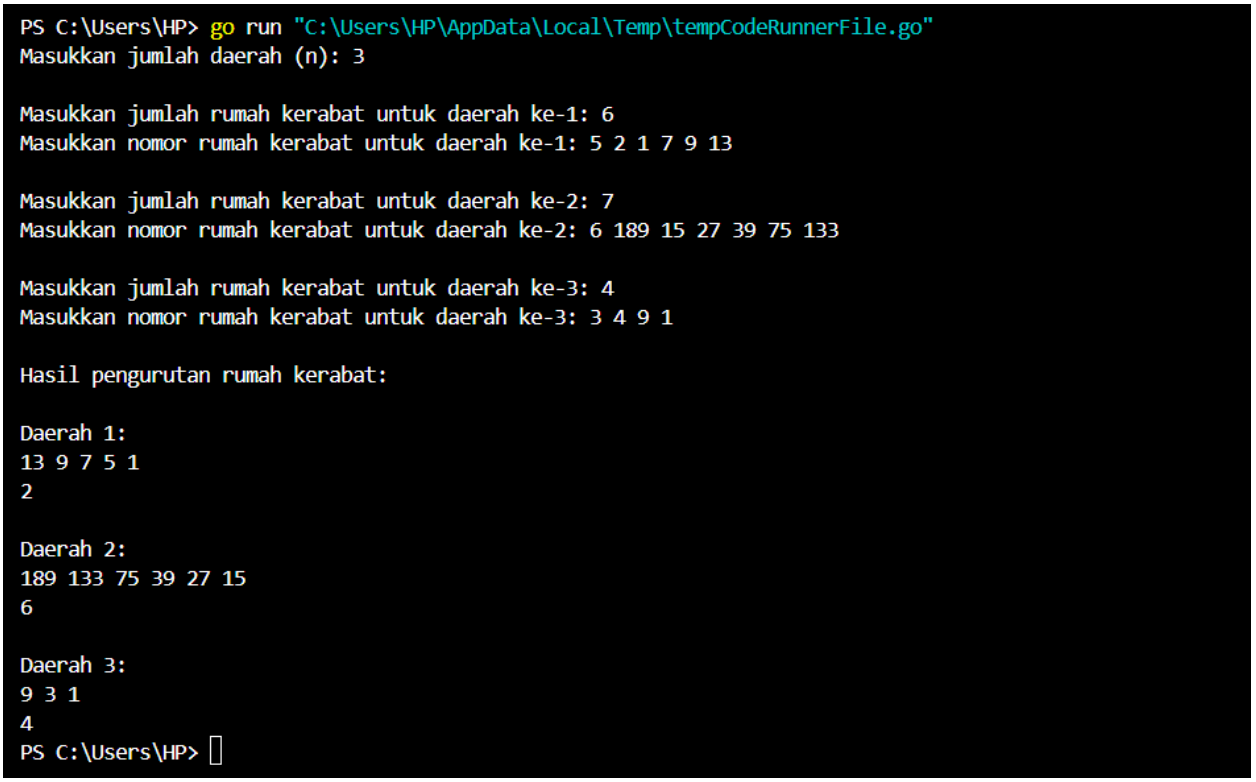
Unguided 1

```
package main

import (
    "fmt"
    "strings"
)

func selectionSortAsc(arr []int) {
    for i := 0; i < len(arr)-1; i++ {
        minIdx := i
        for j := i + 1; j < len(arr); j++ {
            if arr[j] < arr[minIdx] {
                minIdx = j
            }
        }
        arr[i], arr[minIdx] = arr[minIdx],
arr[i]
    }
}

func selectionSortDesc(arr []int) {
    for i := 0; i < len(arr)-1; i++ {
        maxIdx := i
        for j := i + 1; j < len(arr); 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 {
        fmt.Println("Jumlah daerah harus lebih besar dari 0.")
        return
    }

    results := make([]string, 0, n)

    for i := 1; i <= n; i++ {
        var m int
        fmt.Printf("\nMasukkan jumlah rumah kerabat untuk daerah ke-
%d: ", i)
        fmt.Scan(&m)

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

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

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

        selectionSortDesc(ganjil)
        selectionSortAsc(genap)

        ganjilStr := strings.Trim(fmt.Sprint(ganjil), "[]")
        genapStr := strings.Trim(fmt.Sprint(genap), "[]")
```

```
        results = append(results, fmt.Sprintf("%s\n%s", ganjilStr, genapStr))
    }

    fmt.Println("\nHasil pengurutan rumah kerabat:")
    for i, result := range results {
        fmt.Printf("\nDaerah %d:\n%s\n", i+1, result)
    }
}
```

Screenshot



Penjelasan

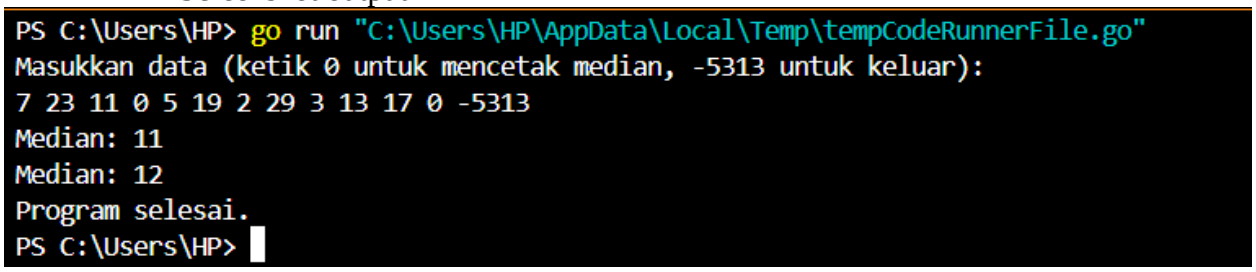
Program ini untuk mengurutkan nomor rumah berdasarkan ganjil dan genap di beberapa daerah. Program meminta pengguna memasukkan jumlah daerah dan jumlah rumah di masing-masing daerah, lalu mengelompokkan nomor rumah menjadi bilangan ganjil dan genap. Bilangan ganjil diurutkan secara menurun (*descending*) menggunakan fungsi selectionSortDesc, sedangkan bilangan genap diurutkan secara menaik (*ascending*) menggunakan fungsi selectionSortAsc. Hasil pengelompokan dan pengurutan ditampilkan dalam format dua baris per daerah: baris pertama untuk bilangan ganjil dan baris kedua untuk bilangan genap. Program memanfaatkan array results untuk menyimpan hasil dari setiap daerah sebelum mencetaknya secara terorganisir.

Unguided 2

```
package main

import "fmt"

func selectionSort(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 printMedian(arr []int) {
    selectionSort(arr)
    n := len(arr)
    if n%2 == 1 {
        fmt.Printf("Median: %d\n", arr[n/2])
    } else {
        mid := n / 2
        median := (arr[mid-1] + arr[mid]) / 2
        fmt.Printf("Median: %d\n", median)
    }
}

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

    fmt.Println("Masukkan data (ketik 0 untuk mencetak median, -
5313 untuk keluar):")
    for {
        fmt.Scan(&input)

        if input == -5313 {
            fmt.Println("Program selesai.")
            break
        }
```

```
        if input == 0 {
            if len(data) == 0 {
                fmt.Println("Data kosong, tidak dapat menghitung
    median.")
            } else {
                printMedian(data)
            }
        } else {
            data = append(data, input)
        }
    }
}
```

Screenshot output



Penjelasan

Program ini berfungsi untuk menghitung median dari sekumpulan data bilangan bulat yang diinputkan pengguna. Median dihitung dengan terlebih dahulu mengurutkan data menggunakan algoritma selection sort. Jika jumlah elemen ganjil, median adalah elemen tengah, sedangkan jika genap, median dihitung sebagai rata-rata dari dua elemen tengah. Program menerima input secara terus-menerus; jika pengguna memasukkan 0, program akan mencetak median dari data yang sudah dimasukkan. Pengguna dapat keluar dari program dengan mengetikkan -5313. Jika data kosong saat median diminta, program akan memberi peringatan bahwa median tidak dapat dihitung. Program memberikan antarmuka sederhana dan interaktif untuk memproses data.

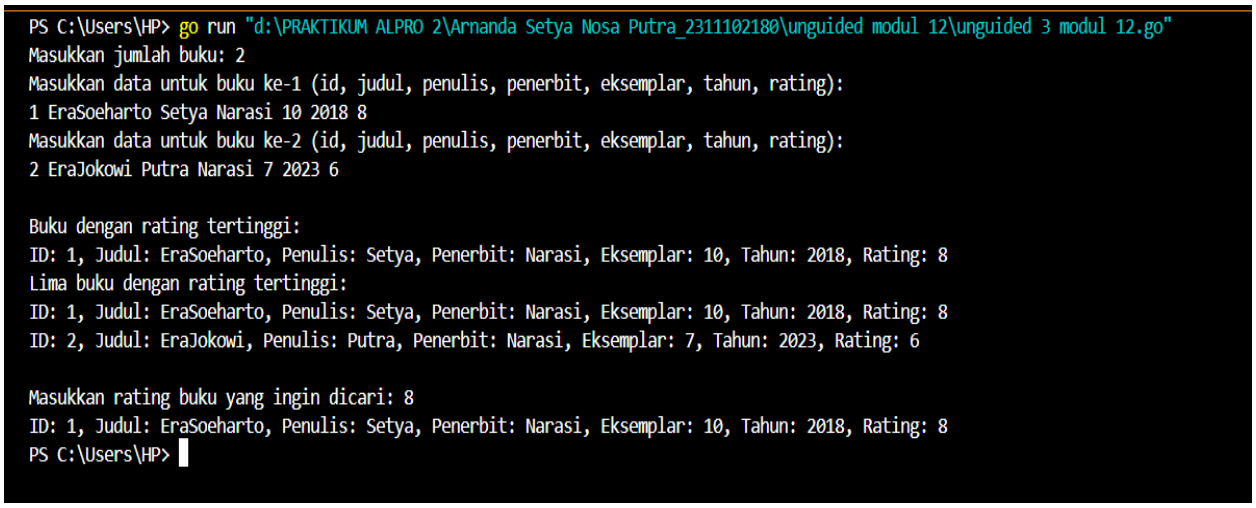
Unguided 3

```
package main

import (
    "fmt"
    "math"
)

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

func insertionSortBooks(pustaka []Buku) {
    n := len(pustaka)
    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 findHighestRatedBook(pustaka []Buku) Buku {
    highest := pustaka[0]
    for _, buku := range pustaka {
        if buku.rating > highest.rating {
            highest = buku
        }
    }
    return highest
}
```

```
func printTop5Books(pustaka []Buku) {
    fmt.Println("Lima buku dengan rating tertinggi:")
    for i := 0; i < int(math.Min(5, float64(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)
    }
}

func searchBooksByRating(pustaka []Buku, rating int) {
    found := false
    for _, buku := range pustaka {
        if buku.rating == rating {
            found = 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 !found {
        fmt.Println("Tidak ada buku dengan rating tersebut.")
    }
}
```

```
func main() {
    var pustaka []Buku
    var n int

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

    if n <= 0 {
        fmt.Println("Jumlah buku harus lebih dari 0.")
        return
    }

    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)
    }

highest := findHighestRatedBook(pustaka)
    fmt.Println("\nBuku dengan rating tertinggi:")
    fmt.Printf("ID: %d, Judul: %s, Penulis: %s, Penerbit: %s, Eksemplar:
    %d, Tahun: %d, Rating: %d\n",
        highest.id, highest.judul, highest.penulis, highest.penerbit,
    highest.eksemplar, highest.tahun, highest.rating)

    insertionSortBooks(pustaka)

    printTop5Books(pustaka)

    var rating int
    fmt.Print("\nMasukkan rating buku yang ingin dicari: ")
    fmt.Scan(&rating)
    searchBooksByRating(pustaka, rating)
}
```

Screenshot output



Penjelasan

Program ini adalah sistem manajemen perpustakaan sederhana yang memungkinkan pengguna untuk mendaftarkan data buku, menemukan buku dengan rating tertinggi, mengurutkan buku berdasarkan rating menggunakan algoritma *insertion sort*, mencetak lima buku dengan rating tertinggi, dan mencari buku berdasarkan rating tertentu. Setiap buku memiliki atribut seperti ID, judul, penulis, penerbit, jumlah eksemplar, tahun terbit, dan rating. Data buku dimasukkan oleh pengguna, dan program memprosesnya untuk menemukan buku dengan rating tertinggi, mengurutkan daftar buku, serta menyediakan fungsi pencarian berdasarkan rating. Program ini memberikan antarmuka interaktif dan hasil yang terstruktur untuk memudahkan pengelolaan data buku.