LAPORAN PRAKTIKUM
ALGORITMA PEMROGRAMAN 2

MODUL 12

PENGURUTAN DATA

Oleh:

CHRIST DANIEL SANTOSO

2311102305

IF-11-02

S1 TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2024

I. DASAR TEORI

Selection Sort

Pengertian:

Selection Sort adalah algoritma pengurutan yang sederhana dan efektif untuk memahami konsep pengurutan data. Algoritma ini bekerja dengan cara mengidentifikasi elemen terkecil (atau terbesar, tergantung kebutuhan) dari bagian array yang belum terurut, lalu menukarnya dengan elemen pada posisi awal bagian tersebut. Proses ini dilakukan secara iteratif, dengan mempersempit bagian yang belum terurut, hingga semua elemen berada dalam urutan yang diinginkan.

Langkah-Langkah Kerja:

1. **Identifikasi Elemen Terkecil**: Mulai dari elemen pertama dalam array, cari elemen terkecil dari seluruh elemen yang belum diurutkan.
2. **Pertukaran Posisi**: Setelah elemen terkecil ditemukan, tukar posisi elemen tersebut dengan elemen pertama di bagian yang belum terurut.
3. **Ulangi Proses**: Lanjutkan ke elemen berikutnya dan ulangi langkah 1 dan 2, hingga seluruh array terurut.

Insertion Sort

Pengertian:

Insertion Sort adalah algoritma pengurutan yang bekerja dengan cara menyusun elemen-elemen dalam array menjadi subarray terurut secara bertahap. Pada setiap iterasi, algoritma mengambil elemen dari bagian yang belum terurut dan menyisipkannya ke dalam posisi yang benar di subarray yang sudah terurut. Proses ini mirip dengan cara seseorang menyusun kartu secara manual, di mana setiap kartu baru ditempatkan pada posisi yang tepat di antara kartu-kartu yang sudah rapi.

Langkah-Langkah Kerja:

1. **Mulai dari Elemen Kedua**: Anggap elemen pertama dari array sudah dalam keadaan terurut.
2. **Ambil Elemen Baru (Key)**: Pilih elemen berikutnya dari bagian yang belum terurut sebagai elemen kunci (key).
3. **Bandingkan dan Geser**: Bandingkan key dengan elemen-elemen di subarray terurut, mulai dari elemen terakhir ke arah awal. Geser elemen yang lebih besar satu posisi ke kanan untuk memberi ruang.
4. **Sisipkan Key**: Tempatkan key pada posisi yang tepat di subarray terurut.
5. **Lanjutkan ke Elemen Berikutnya**: Ulangi proses ini untuk semua elemen dalam array, hingga seluruh array berada dalam urutan yang diinginkan.

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

```
PS C:\Users\cynzw> go run "c:\Users\cynzw\Downloads\GUIDED MODUL 12\GUIDED 1.go"
Masukkan jumlah daerah (n): 2
Masukkan jumlah rumah kerabat untuk daerah ke-1: 3
Masukkan nomor rumah kerabat untuk daerah ke-1: 12 21 22
Hasil urutan rumah untuk daerah ke-1: 22 21 12
Masukkan jumlah rumah kerabat untuk daerah ke-2: 2
Masukkan nomor rumah kerabat untuk daerah ke-2: 21 22
Hasil urutan rumah untuk daerah ke-2: 22 21
PS C:\Users\cynzw>
```

Penjelasan :

Program ini bertujuan untuk mengurutkan nomor rumah di beberapa daerah menggunakan algoritma selection sort. Pengguna akan diminta untuk memasukkan jumlah daerah (n), lalu untuk setiap daerah diminta memasukkan jumlah rumah kerabat (m) beserta daftar nomor rumahnya. Nomor-nomor rumah diurutkan dalam urutan menurun menggunakan selection sort, dan hasilnya ditampilkan untuk setiap daerah. Program juga memeriksa validitas input, memastikan bahwa nilai n dan m berada dalam batas 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 :

```
PS C:\Users\cynzw> go run "c:\Users\cynzw\Downloads\GUIDED MODUL 12\GUIDED 2.go"
Masukkan data (akhiri dengan bilangan negatif):
31 13 25 43 1 7 19 37 -5
Hasil pengurutan: [1 7 13 19 25 31 37 43]
Data berjarak 6
PS C:\Users\cynzw>
```

Penjelasan :

Program ini menggabungkan algoritma pengurutan menggunakan insertion sort dengan pemeriksaan konsistensi jarak antar elemen dalam array. Pengguna diminta untuk memasukkan bilangan bulat satu per satu, dan input akan dihentikan saat bilangan negatif dimasukkan. Setelah data dikumpulkan, program akan mengurutkan angka-angka tersebut menggunakan insertion sort. Kemudian, program memeriksa apakah selisih antar elemen yang sudah terurut tetap konstan. Jika ya, program akan menampilkan informasi bahwa data memiliki jarak yang konsisten beserta nilai selisihnya. Jika tidak, program akan memberi tahu bahwa data tidak memiliki jarak yang tetap.

III. UNGUIDED

Unguided 1

```
package main

import (
    "fmt"
    "strings"
)

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

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

func main() {
    var numRegions int
    fmt.Print("Masukkan jumlah daerah: ")
    fmt.Scan(&numRegions)

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

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

    for region := 1; region <= numRegions; region++ {
        var numHouses int
```

```
        fmt.Printf("\nMasukkan jumlah rumah kerabat untuk daerah ke-%d: ", region)
        fmt.Scan(&numHouses)

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

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

        var oddNumbers, evenNumbers []int
        for _, house := range houses {
            if house%2 == 0 {
                evenNumbers = append(evenNumbers, house)
            } else {
                oddNumbers = append(oddNumbers, house)
            }
        }

        sortDescending(oddNumbers)
        sortAscending(evenNumbers)

        oddString := strings.Trim(fmt.Sprint(oddNumbers), "[]")
        evenString := strings.Trim(fmt.Sprint(evenNumbers), "[]")

        results = append(results, fmt.Sprintf("%s\n%s", oddString, evenString))
    }

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

Screenshot

```
PS C:\Users\cynzw> go run "c:\Users\cynzw\OneDrive\Documents\UNGUIDED MODUL 12\UNGUIDED 1.go"
Masukkan jumlah daerah: 3

Masukkan jumlah rumah kerabat untuk daerah ke-1: 6
Masukkan nomor rumah kerabat untuk daerah ke-1: 5 2 1 7 9 13

Masukkan jumlah rumah kerabat untuk daerah ke-2: 7
Masukkan nomor rumah kerabat untuk daerah ke-2: 6 189 15 27 39 75 133

Masukkan jumlah rumah kerabat untuk daerah ke-3: 4
Masukkan nomor rumah kerabat untuk daerah ke-3: 3 4 9 1

Hasil pengurutan rumah kerabat:

Daerah 1:
13 9 7 5 1
2

Daerah 2:
189 133 75 39 27 15
6

Daerah 3:
9 3 1
4
PS C:\Users\cynzw>
```

Penjelasan

Program ini digunakan untuk mengurutkan nomor rumah berdasarkan kategori ganjil dan genap di berbagai daerah. Pengguna akan diminta untuk memasukkan jumlah daerah serta jumlah rumah di tiap daerah. Program kemudian mengelompokkan nomor rumah ke dalam dua kategori: bilangan ganjil dan genap. Nomor rumah ganjil akan diurutkan secara menurun (descending) menggunakan fungsi selectionSortDesc, sementara nomor rumah genap diurutkan secara menaik (ascending) dengan fungsi selectionSortAsc. Hasil pengelompokan dan pengurutan akan ditampilkan dalam format dua baris per daerah: baris pertama untuk nomor rumah ganjil dan baris kedua untuk nomor rumah genap. Hasil tiap daerah disimpan dalam array results sebelum dicetak secara terstruktur.

Unguided 2

```
package main

import "fmt"

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

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 {
                median := calculateMedian(data)
                fmt.Printf("Median: %d\n", median)
            }
        } else {
            data = append(data, input)
        }
    }
}
```

Screenshot output

```
PS C:\Users\cynzw> go run "c:\Users\cynzw\OneDrive\Documents\UNGUIDED MODUL 12\UNGUIDED 2.go"
Masukkan data (ketik 0 untuk mencetak median, -5313 untuk keluar):
7 23 11 0 5 19 2 29 3 13 17 0 -5313
Median: 11
Median: 12
Program selesai.
PS C:\Users\cynzw>
```

Penjelasan

Program ini dirancang untuk menghitung median dari serangkaian bilangan bulat yang dimasukkan oleh pengguna. Sebelum menghitung median, data akan diurutkan terlebih dahulu menggunakan algoritma selection sort. Jika jumlah data ganjil, median adalah nilai di posisi tengah, sementara jika jumlah data genap, median dihitung sebagai rata-rata dari dua nilai di tengah. Program akan terus menerima input dari pengguna; jika pengguna memasukkan angka 0, program akan menghitung dan menampilkan median dari data yang sudah dimasukkan. Pengguna dapat keluar dari program dengan memasukkan angka -5313. Jika pengguna meminta median ketika data kosong, program akan memberikan pesan peringatan bahwa median tidak dapat dihitung. Program ini memberikan antarmuka yang sederhana dan mudah digunakan untuk memproses data.

Unguided 3

```
package main

import "fmt"

const maxBuku = 7919

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

type KoleksiBuku struct {
   daftar []Buku
}

func TambahkanBuku(koleksi *KoleksiBuku, jumlah int) {
   for i := 0; i < jumlah; 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)
```

```
        if len(koleksi.daftar) < maxBuku {
            koleksi.daftar = append(koleksi.daftar, buku)
        }
    }
}

func TampilkanFavorit(koleksi KoleksiBuku) {
    if len(koleksi.daftar) == 0 {
        fmt.Println("Koleksi kosong.")
        return
    }
    favorit := koleksi.daftar[0]
    for _, buku := range koleksi.daftar {
        if buku.rating > favorit.rating {
            favorit = buku
        }
    }
    fmt.Println("\nBuku dengan rating tertinggi:")
    fmt.Printf("ID: %d, Judul: %s, Penulis: %s, Penerbit: %s, Eksemplar: %d,
Tahun: %d, Rating: %d\n",
        favorit.id, favorit.judul, favorit.penulis, favorit.penerbit, favorit.eksemplar,
favorit.tahun, favorit.rating)
}

func SortirBuku(koleksi *KoleksiBuku) {
    n := len(koleksi.daftar)
    for i := 0; i < n; i++ {
        for j := 0; j < n-i-1; j++ {
            if koleksi.daftar[j].rating < koleksi.daftar[j+1].rating {
                koleksi.daftar[j], koleksi.daftar[j+1] = koleksi.daftar[j+1],
koleksi.daftar[j]
            }
        }
    }
}

func TampilkanTop5(koleksi KoleksiBuku) {
    fmt.Println("\nLima buku dengan rating tertinggi:")
    for i := 0; i < 5 && i < len(koleksi.daftar); i++ {
        buku := koleksi.daftar[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 CariBerdasarkanRating(koleksi KoleksiBuku, rating int) {
    ditemukan := false
    for _, buku := range koleksi.daftar {
```

```
        if buku.rating == rating {
            if !ditemukan {
                fmt.Printf("\nDitemukan buku dengan rating %d:\n", rating)
                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("\nTidak ada buku dengan rating tersebut.")
    }
}

func main() {
    var koleksi KoleksiBuku
    var jumlahBuku int

    fmt.Print("Masukkan jumlah buku di perpustakaan: ")
    fmt.Scan(&jumlahBuku)
    if jumlahBuku <= 0 || jumlahBuku > maxBuku {
        fmt.Println("Jumlah buku harus antara 1 hingga 7919.")
        return
    }

    TambahkanBuku(&koleksi, jumlahBuku)
    TampilkanFavorit(koleksi)
    SortirBuku(&koleksi)
    TampilkanTop5(koleksi)

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

Screenshot output

```
PS C:\Users\cynzw> go run "c:\Users\cynzw\OneDrive\Documents\UNGUIDED MODUL 12\UNGUIDED 3.go"
Masukkan jumlah buku di perpustakaan: 2
Masukkan data untuk buku ke-1 (id, judul, penulis, penerbit, eksemplar, tahun, rating):
1 EraSoeharto Setya Narasi 10 2018 8
Masukkan data untuk buku ke-2 (id, judul, penulis, penerbit, eksemplar, tahun, rating):
2 EraJokowi Putra Narasi 7 2023 6

Buku dengan rating tertinggi:
ID: 1, Judul: EraSoeharto, Penulis: Setya, Penerbit: Narasi, Eksemplar: 10, Tahun: 2018, Rating: 8

Lima buku dengan rating tertinggi:
ID: 1, Judul: EraSoeharto, Penulis: Setya, Penerbit: Narasi, Eksemplar: 10, Tahun: 2018, Rating: 8
ID: 2, Judul: EraJokowi, Penulis: Putra, Penerbit: Narasi, Eksemplar: 7, Tahun: 2023, Rating: 6

Masukkan rating buku yang ingin dicari: 8

Ditemukan buku dengan rating 8:
ID: 1, Judul: EraSoeharto, Penulis: Setya, Penerbit: Narasi, Eksemplar: 10, Tahun: 2018, Rating: 8
PS C:\Users\cynzw>
```

Penjelasan

Program ini adalah aplikasi manajemen perpustakaan sederhana yang memungkinkan pengguna untuk memasukkan data buku, mencari buku dengan rating tertinggi, mengurutkan buku berdasarkan rating menggunakan algoritma insertion sort, menampilkan lima buku teratas berdasarkan rating, serta mencari buku dengan rating tertentu. Setiap buku memiliki atribut seperti ID, judul, penulis, penerbit, jumlah eksemplar, tahun terbit, dan rating. Pengguna memasukkan data buku, dan program akan memprosesnya untuk menemukan buku dengan rating tertinggi, mengurutkan daftar buku, serta menyediakan opsi pencarian berdasarkan rating. Program ini menawarkan antarmuka yang interaktif dan menyajikan hasil dalam format yang terstruktur untuk memudahkan pengelolaan data buku.