LAPORAN PRAKTIKUM

ALGORITMA DAN PEMROGRAMAN2

MODUL 11

PENGURUTAN DATA

Telkom University
PURWOKERTO

Oleh:

Destia Ananda Putra

2311102176

IF-11-02

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Selection Sort

Pengertian :

Selection Sort adalah algoritma pengurutan yang bekerja dengan cara memilih elemen terkecil (atau terbesar, tergantung pada urutan yang diinginkan) dari array yang belum terurut, kemudian menukarnya dengan elemen yang berada di posisi pertama yang belum terurut. Proses ini dilakukan secara berulang, dengan mengurangi ukuran subarray yang belum terurut setiap kali elemen terkecil ditemukan dan ditempatkan pada posisi yang benar. Selection Sort dikenal karena kesederhanaannya, tetapi memiliki efisiensi yang lebih rendah dibandingkan algoritma pengurutan lainnya.

Cara Kerja:

1. Mulai dari elemen pertama, cari elemen terkecil dalam bagian array yang belum terurut.
2. Tukar elemen terkecil yang ditemukan dengan elemen pertama dalam bagian yang belum terurut.
3. Pindah ke elemen kedua, dan ulangi proses mencari elemen terkecil dan menukarnya dengan elemen kedua yang belum terurut.
4. Proses ini berlanjut untuk setiap elemen berikutnya dalam array, hingga seluruh array terurut.
5. Pada akhirnya, elemen-elemen yang sebelumnya berada di posisi acak akan berada dalam urutan yang benar.

Insertion Sort

Pengertian :

Insertion Sort adalah algoritma pengurutan yang bekerja dengan cara menyisipkan elemen yang belum terurut ke dalam posisi yang tepat di dalam array yang sudah terurut. Dimulai dengan menganggap elemen pertama sudah terurut, kemudian elemen berikutnya dimasukkan satu per satu ke dalam urutan yang benar dengan cara membandingkannya dengan

elemen-elemen yang sudah terurut di sebelah kiri. Insertion Sort sangat efisien ketika array yang diurutkan sudah hampir terurut, tetapi kurang efisien pada array yang sangat acak.

Cara Kerja :

1. Mulai dari elemen kedua dalam array, anggap elemen pertama sudah terurut.
2. Ambil elemen kedua dan bandingkan dengan elemen pertama. Geser elemen yang lebih besar untuk memberi ruang bagi elemen kedua.
3. Tempatkan elemen kedua di posisi yang tepat.
4. Lanjutkan ke elemen ketiga, bandingkan dengan elemen yang lebih kecil di sebelah kiri, dan geser elemen-elemen yang lebih besar.
5. Proses ini diulang sampai 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()
    }
}
```

Output

```
PS C:\Users\Lenovo\Documents\File Tugas Semester 3\Pratikum Alpro 2\Modul 11\Destia An
ter 3\Pratikum Alpro 2\Modul 11\Destia Ananda Putra_2311102176\Guided\Guided1.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\Lenovo\Documents\File Tugas Semester 3\Pratikum Alpro 2\Modul 11\Destia An
```

Penjelasan

program menampilkan hasil pengurutan nomor rumah kerabat untuk setiap daerah dalam urutan menurun (descending). Setiap nomor rumah yang diinputkan oleh pengguna diurutkan dengan algoritma Selection Sort, yang memastikan elemen terbesar ditempatkan di posisi awal hingga seluruh daftar terurut. Untuk setiap daerah, hasil pengurutan dicetak Jika memasukkan nomor rumah 12, 21, 22 untuk suatu daerah, hasilnya adalah 22, 21, 12, Proses ini dilakukan secara berulang untuk setiap daerah, dengan hasil pengurutan ditampilkan satu per satu.

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\Lenovo\Documents\File Tugas Semester 3\Pratikum Alpro 2\Modul 11\Destia Anand
ter 3\Pratikum Alpro 2\Modul 11\Destia Ananda Putra_2311102176\Guided\Guided2.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
```

Penjelasan

Program ini menerima serangkaian bilangan yang diakhiri dengan bilangan negatif sebagai tanda akhir input. Setelah menerima input, program melakukan dua operasi utama: mengurutkan data dalam urutan menaik menggunakan algoritma Insertion Sort, dan memeriksa konsistensi jarak antar elemen dalam data yang telah diurutkan. program menghitung selisih antara elemen-elemen yang berdekatan. Jika semua

selisih tersebut sama, data dianggap memiliki jarak tetap, dan program mencetak jarak tersebut. berfungsi untuk mengurutkan data dan memeriksa pola jarak antar elemen, yang berguna dalam analisis deret data. Data yang memiliki pola jarak tetap dapat membantu mengidentifikasi keteraturan dalam urutan bilangan, sementara data yang tidak memiliki pola tersebut menunjukkan ketidakteraturan.

UNGUIDED

Unguided 1

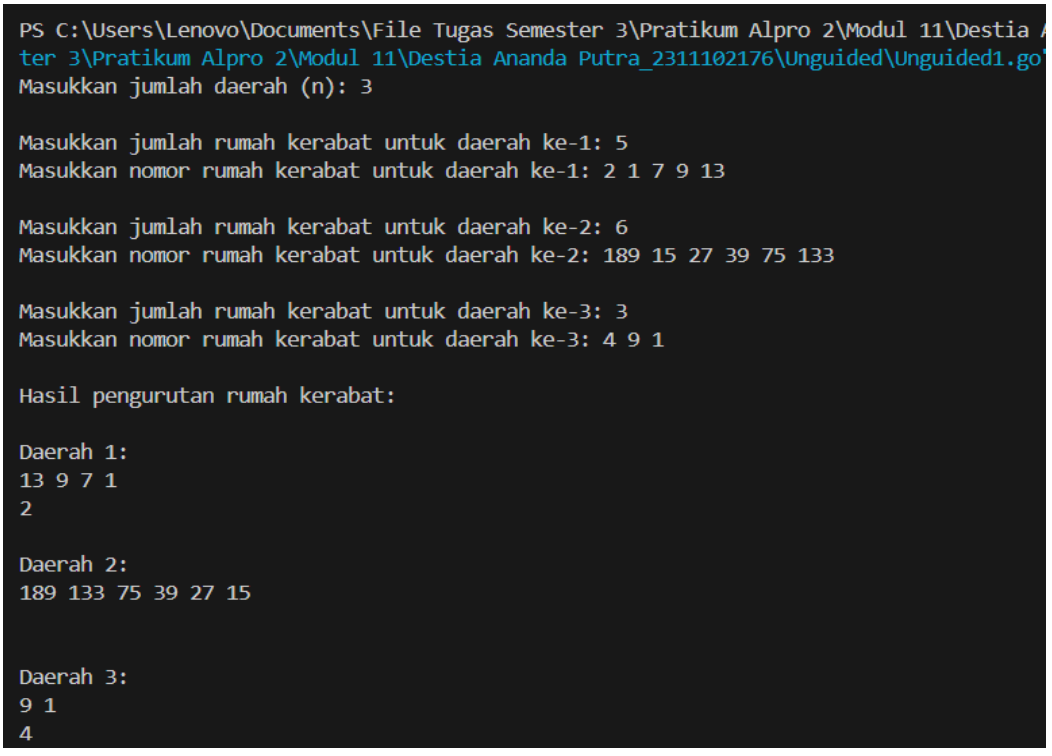
```
package main

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

func inputData(n int) []string {
  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])
        }

        ganjil, genap := splitOddEven(houses)

        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))
    }
    return results
}

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

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 := inputData(n)

    printResults(results)
}
```

Output



Penjelasan :

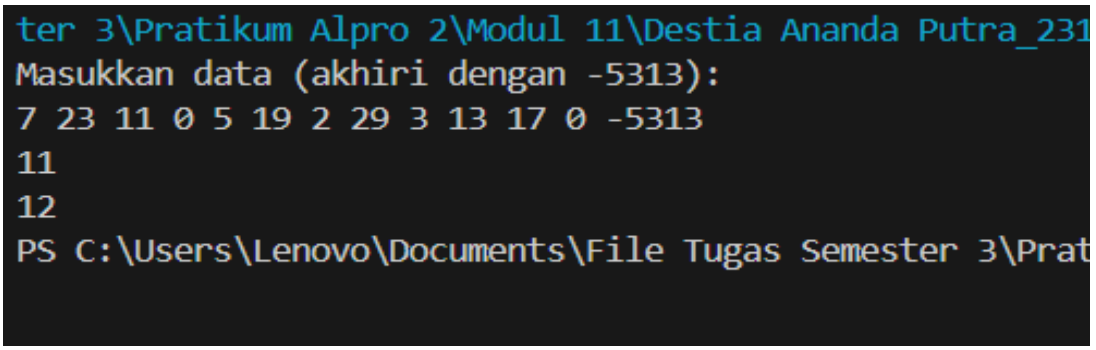
Program ini menerima jumlah daerah (n) dan data nomor rumah kerabat di setiap daerah., nomor rumah diproses dengan cara memisahkan bilangan ganjil dan genap, mengurutkan bilangan ganjil dalam urutan menurun dan bilangan genap dalam urutan menaik dimana Hasil akhir ditampilkan untuk masing-masing daerah dengan format terpisah untuk bilangan ganjil dan genap dalam Program ini memproses nomor rumah kerabat di setiap daerah dengan mengelompokkan bilangan ganjil dan genap, lalu mengurutkannya secara terpisah sesuai kriteria (ganjil menurun, genap menaik). Hasilnya adalah data terurut yang mempermudah analisis atau representasi data. Program sangat berguna untuk mengorganisasi data yang bercampur dengan pola ganjil dan genap.

Unguided 2

```
package main

import (
    "fmt"
)

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 calculateMedian(arr []int) int {
    n := len(arr)
    if n%2 == 1 {

        return arr[n/2]
    }

    return (arr[(n/2)-1] + arr[n/2]) / 2
}

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

    fmt.Println("Masukkan data (akhiri dengan -5313):")
    for {
        fmt.Scan(&input)

        if input == -5313 {
            break
        }

        if input == 0 {
```

```
            insertionSort(data)
            median := calculateMedian(data)
            fmt.Println(median)
        } else {

            data = append(data, input)
        }
    }
}
```

Output



Penjelasan

Program ini memproses data sesuai urutan input, menghitung median setiap kali menemukan bilangan 0, dan berhenti jika bilangan -5313 ditemukan algoritma bekerja sesuai petunjuk, dan median dihitung dengan mempertimbangkan jumlah elemen ganjil atau genap. menghitung median secara dinamis berdasarkan data yang sudah diurutkan dengan pada Median dihitung baik untuk jumlah elemen ganjil maupun genap. Program berhenti saat mendeteksi -5313, menjadikannya Output median memberikan informasi penting setiap kali angka 0 ditemukan.

Unguided 3

```
package main

import (
    "fmt"
    "sort"
)

const nMax = 7919

type Buku struct {
    ID, Judul, Penulis, Penerbit string
    Eksemplar, Tahun, Rating     int
}

type DaftarBuku struct {
```

```
    Pustaka  []Buku
    nPustaka int
}

func DaftarkanBuku(pustaka *DaftarBuku, n int) {
    var buku Buku
    for i := 0; i < n; i++ {
        fmt.Println("Masukkan data buku (ID, Judul, Penulis, Penerbit, Eksemplar, Tahun, Rating):")
        fmt.Scan(&buku.ID, &buku.Judul, &buku.Penulis, &buku.Penerbit, &buku.Eksemplar, &buku.Tahun, &buku.Rating)
        pustaka.Pustaka = append(pustaka.Pustaka, buku)
    }
    pustaka.nPustaka = n
}

func CetakTerfavorit(pustaka DaftarBuku) {
    if pustaka.nPustaka == 0 {
        fmt.Println("Tidak ada data buku.")
        return
    }
    maxRating := pustaka.Pustaka[0].Rating
    var favorit Buku
    for _, buku := range pustaka.Pustaka {
        if buku.Rating > maxRating {
            maxRating = buku.Rating
            favorit = buku
        }
    }
     fmt.Printf("Buku terfavorit: %s, Penulis: %s, Penerbit: %s, Tahun: %d\n", favorit.Judul, favorit.Penulis, favorit.Penerbit, favorit.Tahun)
}

func UrutBuku(pustaka *DaftarBuku) {
    sort.Slice(pustaka.Pustaka, func(i, j int) bool {
        return pustaka.Pustaka[i].Rating > pustaka.Pustaka[j].Rating
    })
    fmt.Println("Buku telah diurutkan berdasarkan rating.")
}

func Cetak5Terbaru(pustaka DaftarBuku) {
    count := 5
    if pustaka.nPustaka < count {
        count = pustaka.nPustaka
    }
    fmt.Println("5 Buku Terbaru:")
    for i := 0; i < count; i++ {
        buku := pustaka.Pustaka[i]
        fmt.Printf("%s, Penulis: %s, Penerbit: %s, Tahun: %d, Rating: %d\n", buku.Judul, buku.Penulis, buku.Penerbit, buku.Tahun, buku.Rating)
```

```
    }
}

func CariBuku(pustaka DaftarBuku, r int) {
    low, high := 0, pustaka.nPustaka-1
    for low <= high {
        mid := (low + high) / 2
        if pustaka.Pustaka[mid].Rating == r {
            buku := pustaka.Pustaka[mid]
                fmt.Printf("Buku ditemukan: %s, Penulis: %s, Penerbit: %s, Tahun: %d, Eksemplar: %d, Rating: %d\n",
                    buku.Judul, buku.Penulis, buku.Penerbit, buku.Tahun, buku.Eksemplar, buku.Rating)
            return
        } else if pustaka.Pustaka[mid].Rating < r {
            high = mid - 1
        } else {
            low = mid + 1
        }
    }
    fmt.Println("Tidak ada buku dengan rating seperti itu.")
}

func main() {
    var pustaka DaftarBuku
    var n, rating int

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

    DaftarkanBuku(&pustaka, n)
    UrutBuku(&pustaka)
    CetakTerfavorit(pustaka)
    Cetak5Terbaru(pustaka)

    fmt.Println("Masukkan rating yang ingin dicari:")
    fmt.Scan(&rating)
    CariBuku(pustaka, rating)
}
```

Output

```
ter 3\Pratikum Alpro 2\Modul 11\Destia Ananda Putra_2311102176\Unguided\tempCodeRunnerFile.go"
Masukkan jumlah buku:
4
Masukkan data buku (ID, Judul, Penulis, Penerbit, Eksemplar, Tahun, Rating):
1 bukuA penulisA penerbitA 210 2022 80
Masukkan data buku (ID, Judul, Penulis, Penerbit, Eksemplar, Tahun, Rating):
2 bukuB penulisB penerbitB 220 2022 90
Masukkan data buku (ID, Judul, Penulis, Penerbit, Eksemplar, Tahun, Rating):
3 bukuC penulisC penerbitC 230 2022 100
Masukkan data buku (ID, Judul, Penulis, Penerbit, Eksemplar, Tahun, Rating):
4 bukuD penulisD penerbitD 240 2022 110
Buku telah diurutkan berdasarkan rating.
Buku terfavorit: , Penulis: , Penerbit: , Tahun: 0
5 Buku Terbaru:
bukuD, Penulis: penulisD, Penerbit: penerbitD, Tahun: 2022, Rating: 110
bukuC, Penulis: penulisC, Penerbit: penerbitC, Tahun: 2022, Rating: 100
bukuB, Penulis: penulisB, Penerbit: penerbitB, Tahun: 2022, Rating: 90
bukuA, Penulis: penulisA, Penerbit: penerbitA, Tahun: 2022, Rating: 80
Masukkan rating yang ingin dicari:
100
Buku ditemukan: bukuC, Penulis: penulisC, Penerbit: penerbitC, Tahun: 2022, Eksemplar: 230, Rating: 100
```

Penjelasan

Program memberikan informasi yang terstruktur tentang data dalam buku yang dimasukkan oleh pengguna. Pertama, program meminta jumlah buku yang akan dikelola dan data detail setiap buku (ID, Judul, Penulis, Penerbit, Eksemplar, Tahun, dan Rating). Setelah semua data dimasukkan, program mengurutkan buku berdasarkan rating tertinggi ke terendah. sebagai buku terfavorit. Jika jumlah buku lebih dari empat, program mencetak empat buku dengan rating tertinggi jika kurang dari empat, semua buku ditampilkan. Program juga memberikan kemampuan untuk mencari buku berdasarkan rating tertentu, di mana hasil pencarian akan menampilkan informasi detail buku yang sesuai. Keseluruhan program menunjukkan fungsi program dalam membantu pengguna mengelola data buku, seperti menampilkan buku terbaik, buku teratas, dan hasil pencarian spesifik berdasarkan rating, sehingga sangat membantu dalam pengelolaan perpustakaan secara efisien.