

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

MODUL I2

PENGURURUTAN DATA



Oleh:

FAJAR FARIZQI AZMI

2311102192

IF-11-02

**S1 TEKNIK INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO**

2024

Dasar Teori

SelectionSort

Selection Sort adalah salah satu algoritma pengurutan yang sederhana dan mudah dipahami.

Deskripsi

Selection Sort bekerja dengan cara membagi array menjadi dua bagian: bagian yang sudah terurut dan bagian yang belum terurut. Proses pengurutan dilakukan dengan cara memilih elemen terkecil dari bagian yang belum terurut dan menukarnya dengan elemen pertama dari bagian yang belum terurut. Proses ini diulang hingga seluruh array terurut.

Langkah-langkah Algoritma

1. **Inisialisasi:** Mulai dari elemen pertama dalam array.
2. **Pilih Elemen Terkecil:** Temukan elemen terkecil dalam bagian yang belum terurut.
3. **Tukar Elemen:** Tukar elemen terkecil yang ditemukan dengan elemen pertama dari bagian yang belum terurut.
4. **Pindah ke Elemen Berikutnya:** Pindah ke elemen berikutnya dan ulangi proses hingga seluruh array terurut.

Contoh

Misalkan kita memiliki array: [64, 25, 12, 22, 11].

1. Pada langkah pertama, elemen terkecil adalah 11. Tukar 11 dengan 64: [11, 25, 12, 22, 64].
2. Pada langkah kedua, elemen terkecil dari [25, 12, 22, 64] adalah 12. Tukar 12 dengan 25: [11, 12, 25, 22, 64].
3. Ulangi hingga array terurut: [11, 12, 22, 25, 64].

Kelebihan dan Kekurangan

- **Kelebihan:**
 - Mudah dipahami dan diimplementasikan.
 - Tidak memerlukan ruang tambahan (in-place sorting).
- **Kekurangan:**

- Memiliki kompleksitas waktu $O(n^2)$, yang membuatnya tidak efisien untuk array besar.
- Tidak stabil, yang berarti tidak mempertahankan urutan elemen yang sama.

Kesimpulan

Selection Sort adalah algoritma yang bermanfaat untuk belajar tentang konsep pengurutan, meskipun tidak efisien untuk penggunaan dalam skala besar. Algoritma ini lebih cocok untuk data kecil atau sebagai pengantar untuk algoritma pengurutan yang lebih kompleks.

InsertionSort

Insertion Sort adalah algoritma pengurutan yang sederhana dan efisien untuk daftar kecil. Algoritma ini membangun urutan akhir secara bertahap dengan cara memasukkan elemen satu per satu ke dalam posisi yang tepat.

Deskripsi

Insertion Sort bekerja dengan menganggap bahwa bagian awal dari array sudah terurut. Kemudian, elemen berikutnya diambil dan dimasukkan ke dalam bagian yang terurut dengan cara membandingkannya dengan elemen-elemen dalam bagian tersebut.

Langkah-langkah Algoritma

1. **Inisialisasi:** Mulai dari elemen kedua dalam array (indeks 1), anggap elemen pertama (indeks 0) sudah terurut.
2. **Ambil Elemen:** Ambil elemen yang akan dimasukkan (misalnya, elemen saat ini).
3. **Bandingkan dan Geser:** Bandingkan elemen yang diambil dengan elemen-elemen dalam bagian yang terurut. Geser elemen yang lebih besar ke kanan untuk memberi ruang pada elemen yang diambil.
4. **Masukkan Elemen:** Tempatkan elemen yang diambil pada posisi yang tepat.
5. **Ulangi:** Ulangi proses untuk semua elemen dalam array sampai seluruh array terurut.

Contoh

Misalkan kita memiliki array: [5, 2, 9, 1, 5, 6].

1. Mulai dari elemen kedua (2), bandingkan dengan 5. Karena 2 lebih kecil, geser 5 ke kanan dan masukkan 2: [2, 5, 9, 1, 5, 6].

2. Ambil elemen berikutnya (9). Karena 9 sudah lebih besar dari 5, tidak perlu menggeser. Array tetap: [2, 5, 9, 1, 5, 6].
3. Ambil 1, geser 9, 5, dan 2 ke kanan, lalu masukkan 1: [1, 2, 5, 9, 5, 6].
4. Ambil 5 dan masukkan di posisi yang tepat: [1, 2, 5, 5, 9, 6].
5. Terakhir, ambil 6, geser 9 dan masukkan 6: [1, 2, 5, 5, 6, 9].

Kelebihan dan Kekurangan

- **Kelebihan:**
 - Mudah dipahami dan diimplementasikan.
 - Efisien untuk daftar kecil atau hampir terurut.
 - Stabil, mempertahankan urutan elemen yang sama.
- **Kekurangan:**
 - Memiliki kompleksitas waktu $O(n^2)$ untuk daftar besar.
 - Tidak efisien untuk data yang besar jika dibandingkan dengan algoritma lain seperti Quick Sort atau Merge Sort.

Kesimpulan

Insertion Sort adalah algoritma yang baik untuk pengurutan dalam situasi tertentu, terutama ketika bekerja dengan daftar kecil atau data yang sudah hampir terurut. Meskipun tidak seefisien algoritma pengurutan lainnya pada skala besar, Insertion Sort memberikan pemahaman yang baik tentang konsep dasar pengurutan.

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)

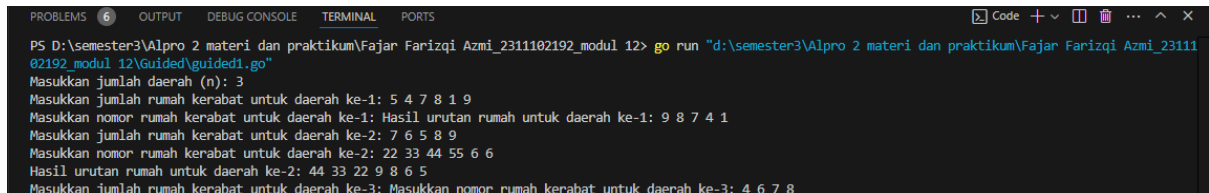
}

fmt.Println()

}

}
```

Screenshot output :



```
PS D:\semester3\Alpro 2 materi dan praktikum\Fajar Farizqi Azmi_2311102192_modul 12> go run "d:\semester3\Alpro 2 materi dan praktikum\Fajar Farizqi Azmi_2311102192_modul 12\Guided\guided1.go"
Masukkan jumlah daerah (n): 3
Masukkan jumlah rumah kerabat untuk daerah ke-1: 5 4 7 8 1 9
Masukkan nomor rumah kerabat untuk daerah ke-1: Hasil urutan rumah untuk daerah ke-1: 9 8 7 4 1
Masukkan jumlah rumah kerabat untuk daerah ke-2: 7 6 5 8 9
Masukkan nomor rumah kerabat untuk daerah ke-2: 22 33 44 55 6 6
Masukkan jumlah rumah kerabat untuk daerah ke-3: 4 6 7 8
Masukkan nomor rumah kerabat untuk daerah ke-3: 4 6 7 8
```

Deskripsi program :

Program ini adalah untuk mengurutkan nomor rumah kerabat di beberapa daerah secara menurun menggunakan algoritma Selection Sort. Program menerima input jumlah daerah, jumlah rumah di setiap daerah, dan nomor rumahnya, kemudian menampilkan hasil pengurutan untuk setiap daerah. Program ini dilengkapi validasi input untuk memastikan data yang dimasukkan sesuai.

Guided 2

Source code :

```
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)

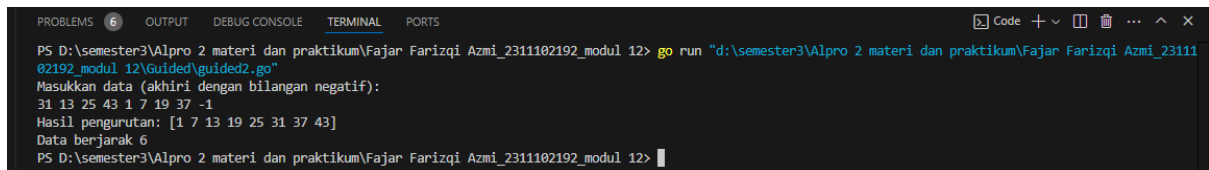
} else {

    fmt.Println("Data berjarak tidak tetap")

}

}
```

Screenshot output :



```
PROBLEMS 6 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\semester3\Alpro 2 materi dan praktikum\Fajar Farizqi Azmi_2311102192_modul 12> go run "d:\semester3\Alpro 2 materi dan praktikum\Fajar Farizqi Azmi_2311102192_modul 12\Guided\guided2.go"
Masukkan data (akhiri dengan bilangan negatif):
31 13 25 43 1 7 19 37 -1
Hasil pengurutan: [1 7 13 19 25 31 37 43]
Data berjarak 6
PS D:\semester3\Alpro 2 materi dan praktikum\Fajar Farizqi Azmi_2311102192_modul 12>
```

Deskripsi program :

program ini dibuat untuk mengurutkan data angka yang dimasukkan pengguna dan memeriksa apakah data tersebut memiliki pola jarak tetap antar elemen setelah diurutkan. Program ini berguna untuk analisis data dengan pola linier atau untuk memastikan konsistensi jarak antar nilai dalam sebuah dataset.

UNGUIDED

Unguided 1

Source code :

```
package main

import "fmt"

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] {

                minIdx = j

            }

        }

        arr[i], arr[minIdx] = arr[minIdx], arr[i]

    }

}
```

```
func main() {  
  
    var n int  
  
    fmt.Println("Masukan")  
  
    fmt.Scan(&n)  
  
  
    if n <= 0 || n >= 1000 {  
  
        fmt.Println("Jumlah daerah harus di antara 1 dan 999.")  
  
        return  
    }  
  
  
    masukan := make([]int, n)  
  
    for i := 0; i < n; i++ {  
  
        var m int  
  
        fmt.Scan(&m)  
  
  
        if m <= 0 || m >= 1000000 {  
  
            fmt.Println("Jumlah rumah harus di antara 1 dan 999999.")  
  
            return  
        }  
    }  
}
```

```
masukan[i] = make([]int, m)
```

```
for j := 0; j < m; j++ {
```

```
    fmt.Scan(&masukan[i][j])
```

```
}
```

```
}
```

```
fmt.Println("\nKeluaran:")
```

```
for _, daerah := range masukan {
```

```
    var ganjil []int
```

```
    var genap []int
```

```
    for _, num := range daerah {
```

```
        if num%2 == 0 {
```

```
            genap = append(genap, num)
```

```
        } else {
```

```
            ganjil = append(ganjil, num)
```

```
        }
```

```
    }
```

```
selectionSortAsc(ganjil)
```

```
selectionSortAsc(genap)

hasil := append(ganjil, genap...)

for _, val := range hasil {

    fmt.Print(val, " ")

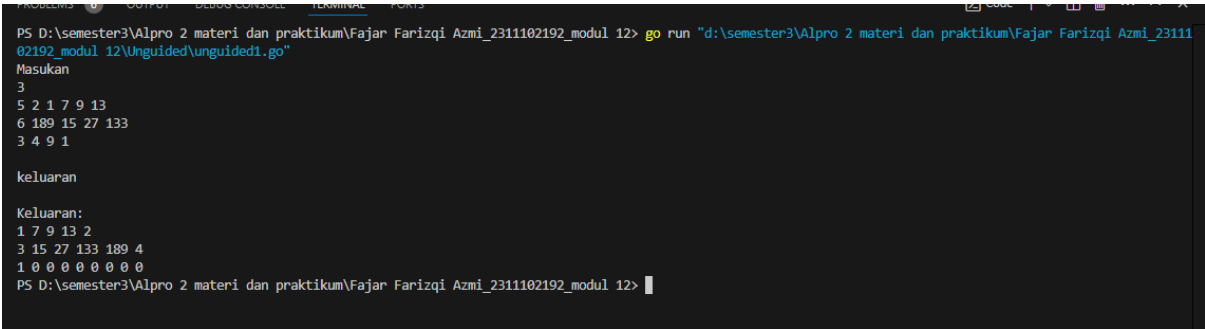
}

fmt.Println()

}

}
```

Screenshot output :



Deskripsi program :

program ini digunakan untuk mengurutkan nomor rumah di tiap daerah dengan memisahkan angka ganjil dan genap, mengurutkan masing-masing kelompok secara menaik, lalu menampilkan nomor rumah dengan urutan ganjil terlebih dahulu diikuti oleh genap.

Unguided 2

Source code :

```
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 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 input int
    var data []int

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

```

        if input == -5313 {
            break
        }

        if input == 0 {
            if len(data) == 0 {
                continue
            }
            selectionSort(data)
            median := calculateMedian(data)
            fmt.Println(median)
        } else {
            data = append(data, input)
        }
    }
}

```

Screenshot output :

```

PS D:\semester3\Alpro 2 materi dan praktikum\Fajar Farizqi Azmi_2311102192_modul 12> go run "d:\semester3\Alpro 2 materi dan praktikum\Fajar Farizqi Azmi_2311102192_modul 12\Unguided\unguided2.go"
7 23 11 0 5 19 2 29 3 13 17 0 -5313
11
12
PS D:\semester3\Alpro 2 materi dan praktikum\Fajar Farizqi Azmi_2311102192_modul 12>

```

Deskripsi program :

Program di atas adalah aplikasi Go untuk menghitung median dari serangkaian angka yang dimasukkan pengguna secara interaktif. Program memungkinkan pengguna untuk memasukkan angka secara bertahap, menghitung median setiap kali angka 0 dimasukkan, dan menghentikan program dengan memasukkan angka -5313.

Unguided 3

Sourch code :

```

package main

```

```
import (  
    "fmt"  
)  
  
const nMax = 7919  
  
type Buku struct {  
    id    int  
    judul string  
    penulis string  
    penerbit string  
    eksemplar int  
    tahun  int  
    rating int  
}  
  
type DaftarBuku struct {  
    pustaka []Buku  
}  
  
func DaftarkanBuku(pustaka *DaftarBuku, buku Buku) {  
    if len(pustaka.pustaka) < nMax {  
        pustaka.pustaka = append(pustaka.pustaka, buku)  
    }  
}  
  
func CetakTerfavorit(pustaka DaftarBuku, n int) {  
    if n == 0 {  
        fmt.Println("Tidak ada buku dalam pustaka.")  
        return  
    }  
}
```

```

    terfavorit := pustaka.pustaka[0]
    for i := 1; i < n; i++ {
        if pustaka.pustaka[i].rating > terfavorit.rating {
            terfavorit = pustaka.pustaka[i]
        }
    }

    fmt.Println("Buku Terfavorit:")
    fmt.Printf("Judul   : %s\nPenulis : %s\nPenerbit : %s\nTahun   : %d\nRating  : %d\n\n",
        terfavorit.judul, terfavorit.penulis, terfavorit.penerbit, terfavorit.tahun,
        terfavorit.rating)
}

func UrutBuku(pustaka *DaftarBuku, n int) {
    for i := 1; i < n; i++ {
        key := pustaka.pustaka[i]
        j := i - 1
        for j >= 0 && pustaka.pustaka[j].rating < key.rating {
            pustaka.pustaka[j+1] = pustaka.pustaka[j]
            j--
        }
        pustaka.pustaka[j+1] = key
    }
}

func Cetak5Terbaru(pustaka DaftarBuku, n int) {
    if n == 0 {
        fmt.Println("Tidak ada buku dalam pustaka.")
        return
    }

    fmt.Println("3 Buku Dengan Rating Tertinggi:")
    for i := 0; i < 5 && i < n; i++ {

```

```

        buku := pustaka.pustaka[i]
        fmt.Printf("Judul : %s\nRating : %d\n", buku.judul, buku.rating)
    }
}

func CariBuku(pustaka DaftarBuku, n, r int) {
    UrutBuku(&pustaka, n)

    low, high := 0, n-1
    for low <= high {
        mid := (low + high) / 2
        if pustaka.pustaka[mid].rating == r {
            buku := pustaka.pustaka[mid]
            fmt.Printf("Buku ditemukan:\nJudul   : %s\nPenulis : %s\nPenerbit : %s\nTahun   : %d\nEksemplar: %d\nRating  : %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 int

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

```

```

for i := 0; i < n; i++ {
    var id, eksemplar, tahun, rating int
    var judul, penulis, penerbit string

    fmt.Printf("Masukkan data buku ke-%d:\n", i+1)
    fmt.Print("ID      : ")
    fmt.Scanln(&id)
    fmt.Print("Judul   : ")
    fmt.Scanln(&judul)
    fmt.Print("Penulis  : ")
    fmt.Scanln(&penulis)
    fmt.Print("Penerbit : ")
    fmt.Scanln(&penerbit)
    fmt.Print("Eksemplar: ")
    fmt.Scanln(&eksemplar)
    fmt.Print("Tahun    : ")
    fmt.Scanln(&tahun)
    fmt.Print("Rating   : ")
    fmt.Scanln(&rating)

    buku := Buku{id, judul, penulis, penerbit, eksemplar, tahun, rating}
    DaftarkanBuku(&pustaka, buku)
    fmt.Println()
}

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

var targetRating int
fmt.Print("\nMasukkan rating buku yang ingin dicari: ")
fmt.Scanln(&targetRating)
CariBuku(pustaka, n, targetRating)

```

```
}
```

Screenshot output :

```
PS D:\semester3\Alpro 2 materi dan praktikum\Fajar Farizqi Azmi_2311102192_modul 12> go run "d:\semester3\Alpro 2 materi dan praktikum\Fajar Farizqi Azmi_2311102192_modul 12\Unguided\unguided3.go"
Masukkan jumlah buku: 3
Masukkan data buku ke-1:
ID      : 200
Judul   : terserah
Penulis : an
Penerbit : en
Eksemplar: 100
Tahun   : 1988
Rating  : 89

Masukkan data buku ke-2:
ID      : 201
Judul   : jokmotor
Penulis : ak
Penerbit : ab
Eksemplar: 101
Tahun   : 2000
Rating  : 84

Masukkan data buku ke-3:
ID      : 202
Judul   : jekime
Penulis : oo
Penerbit : po
Eksemplar: 102
Tahun   : 2004
Rating  : 99
```

```
Buku Terfavorit:
Judul   : jekime
Penulis : oo
Penerbit : po
Tahun   : 2004
Rating  : 99

3 Buku Dengan Rating Tertinggi:
Judul   : jekime
Rating  : 99
Judul   : terserah
Rating  : 89
Judul   : jokmotor
Rating  : 84

Masukkan rating buku yang ingin dicari: 99
Buku ditemukan:
Judul   : jekime
Penulis : oo
Penerbit : po
Tahun   : 2004
Eksemplar: 102
Rating  : 99
PS D:\semester3\Alpro 2 materi dan praktikum\Fajar Farizqi Azmi_2311102192_modul 12> 
```

Deskripsi program :

Program ini dirancang sebagai sistem sederhana untuk mengelola data buku dalam sebuah perpustakaan digital. Program ini memungkinkan pengguna untuk menambahkan data buku secara manual, menyimpannya dalam struktur data, dan melakukan berbagai operasi seperti mengurutkan buku berdasarkan rating, mencari buku berdasarkan rating tertentu, serta

menampilkan buku dengan rating tertinggi. Data buku yang dikelola meliputi ID, judul, penulis, penerbit, jumlah eksemplar, tahun terbit, dan rating. Program ini menggunakan algoritma insertion sort untuk mengurutkan data buku berdasarkan rating dan algoritma binary search untuk mencari buku secara efisien. Dengan fitur-fitur ini, program ini dapat membantu dalam mengorganisasi dan mengelola informasi tentang koleksi buku dalam sebuah perpustakaan.