

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2
MODUL 12&13
PENGURUTAN DATA**



Oleh:

MUHAMMAD FAUZAN

103112400064

12 IF 01

**S1 TEKNIK INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025**

I. DASAR TEORI

1. Ide Algoritma Selection Sort

Selection Sort adalah algoritma pengurutan yang bekerja dengan cara mencari elemen terkecil (atau terbesar) dalam rentang data yang belum terurut, kemudian menukarnya dengan elemen pertama dari rentang tersebut. Proses ini diulang hingga seluruh data terurut. Langkah-langkah algoritma ini adalah:

1. Cari nilai terkecil dalam rentang data yang belum terurut.
2. Tukar posisi elemen terkecil dengan elemen pertama yang belum terurut.
3. Ulangi proses ini hingga seluruh elemen terurut.

Algoritma ini memerlukan dua proses utama: pencarian nilai ekstrim dan pertukaran nilai (swap). Selection Sort memiliki kompleksitas waktu $O(n^2)$, sehingga kurang efisien untuk dataset yang sangat besar.

2. Ide Algoritma Insertion Sort

Insertion Sort mengurutkan data dengan cara menyisipkan elemen yang belum terurut ke dalam posisi yang sesuai di dalam data yang sudah terurut. Berbeda dengan Selection Sort, Insertion Sort tidak mencari nilai ekstrim, melainkan langsung mencari posisi yang tepat untuk elemen tersebut. Langkah-langkah algoritma ini adalah:

1. Ambil elemen yang belum terurut dan cari posisi yang tepat dalam data yang sudah terurut.
2. Geser elemen-elemen yang lebih besar untuk memberi ruang bagi elemen yang disisipkan.
3. Ulangi proses ini hingga seluruh data terurut.

Insertion Sort bekerja dengan prinsip pencarian sekuensial dan penyisipan elemen ke dalam urutan yang sudah ada. Algoritma ini memiliki kompleksitas waktu $O(n^2)$ pada kasus terburuk, namun dapat lebih efisien untuk dataset yang kecil atau hampir terurut.

II. GUIDED

Source Code + Screenshot hasil program beserta penjelasan

Contoh 1

```
package main

import (
    "fmt"
)

type Mahasiswa struct {
    Nama string
    Nilai int
}

func printData(mahasiswas []Mahasiswa) {
    for _, m := range mahasiswas {
        fmt.Printf("%s: %d", m.Nama, m.Nilai)
    }
    fmt.Println()
}

func selectionSort(mahasiswas []Mahasiswa) {
    n := len(mahasiswas)
    fmt.Println("\nProses Selection Sort:")
    for i := 0; i < n-1; i++ {
        minIdx := i
        for j := i + 1; j < n; j++ {
            if mahasiswas[j].Nilai < mahasiswas[minIdx].Nilai {
                minIdx = j
            }
        }

        mahasiswas[i], mahasiswas[minIdx] = mahasiswas[minIdx],
mahasiswas[i]
        fmt.Printf("Iterasi ke-%d: ", i+1)
        printData(mahasiswas)
    }
}

func main() {
    data := []Mahasiswa{
```

```

        {" Budi", 75},
        {" Ani", 90},
        {" Dedi", 65},
        {" Citra", 85},
        {" Eka", 70},
    }
    fmt.Println("Data Sebelum Selection Sort:")
    printData(data)
    selectionSort(data)
    fmt.Println("\ndata setelah Selection Sort (Berdasarkan Nilai):")
    printData(data)
}

```

Screenshots Output

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\CAN\kuliah\ZAN (Directory VSCODE)> go run "c:\CAN\kuliah\ZAN (Directory VSCODE)\ALGORITMA 2\2400064_Guided1.go"
• Data Sebelum Selection Sort:
  Budi: 75 Ani: 90 Dedi: 65 Citra: 85 Eka: 70

Proses Selection Sort:
Iterasi ke-1: Dedi: 65 Ani: 90 Budi: 75 Citra: 85 Eka: 70
Iterasi ke-2: Dedi: 65 Eka: 70 Budi: 75 Citra: 85 Ani: 90
Iterasi ke-3: Dedi: 65 Eka: 70 Budi: 75 Citra: 85 Ani: 90
Iterasi ke-4: Dedi: 65 Eka: 70 Budi: 75 Citra: 85 Ani: 90

data setelah Selection Sort (Berdasarkan Nilai):
Dedi: 65 Eka: 70 Budi: 75 Citra: 85 Ani: 90

```

// Foto hasil dari menjalankan code

Deskripsi: Program ini dibuat untuk mengurutkan data mahasiswa berdasarkan nilai dari yang terendah ke tertinggi menggunakan algoritma selection sort. Data mahasiswa disimpan dalam bentuk slice struct, yang masing-masing berisi nama dan nilai. Program menampilkan data sebelum dan sesudah proses pengurutan. Selain itu, setiap iterasi dalam proses selection sort juga ditampilkan secara rinci agar pengguna dapat memahami bagaimana perubahan data terjadi pada setiap langkah.

Contoh 2

```
package main

import "fmt"

type Mahasiswa struct {
    Nama string
    Nilai int
}

func printData(mahasiswas []Mahasiswa) {
    for _, m := range mahasiswas {
        fmt.Printf("%s: %d ", m.Nama, m.Nilai)
    }
    fmt.Println()
}

func insertionSort(mahasiswas []Mahasiswa) {
    fmt.Println("\nProses Insertion Sort:")
    for i := 1; i < len(mahasiswas); i++ {
        key := mahasiswas[i]
        j := i - 1

        for j >= 0 && mahasiswas[j].Nilai > key.Nilai {
            mahasiswas[j+1] = mahasiswas[j]
            j--
        }
        mahasiswas[j+1] = key

        // Tampilkan kondisi array setelah iterasi
        fmt.Printf("Iterasi ke-%d: ", i)
        printData(mahasiswas)
    }
}

func main() {
    data := []Mahasiswa{
        {"Budi", 75},
        {"Ani", 90},
        {"Dedi", 65},
        {"Citra", 85},
        {"Eka", 70},
    }
}
```

```

    fmt.Println("Data sebelum Insertion Sort:")
    printData(data)

    insertionSort(data)

    fmt.Println("\nData setelah Insertion Sort (berdasarkan Nilai):")
    printData(data)
}

```

Screenshots Output

```

PS C:\CAN\Kuliah\ZAN (Directory VSCODE)> go run "c:\CAN\Kuliah\ZAN (Directory VSCODE)\ALGORITMA 2\2400064_Guided2.go"
Data sebelum Insertion Sort:
Budi: 75 Ani: 90 Dedi: 65 Citra: 85 Eka: 70

Proses Insertion Sort:
Iterasi ke-1: Budi: 75 Ani: 90 Dedi: 65 Citra: 85 Eka: 70
Iterasi ke-2: Dedi: 65 Budi: 75 Ani: 90 Citra: 85 Eka: 70
Iterasi ke-3: Dedi: 65 Budi: 75 Citra: 85 Ani: 90 Eka: 70
Iterasi ke-4: Dedi: 65 Eka: 70 Budi: 75 Citra: 85 Ani: 90

Data setelah Insertion Sort (berdasarkan Nilai):
Dedi: 65 Eka: 70 Budi: 75 Citra: 85 Ani: 90

```

// Foto hasil dari menjalankan code

Deskripsi: Program ini berfungsi untuk mengurutkan data mahasiswa berdasarkan nilai menggunakan algoritma insertion sort. Setiap mahasiswa diwakili oleh struct yang menyimpan nama dan nilai. Program akan menampilkan data awal sebelum proses pengurutan, kemudian menampilkan kondisi data setelah setiap iterasi selama proses insertion sort. Setelah proses selesai, data yang telah terurut dari nilai terendah ke tertinggi ditampilkan kembali ke layar.

III. UNGUIDED

Source Code + Screenshot hasil program beserta penjelasan

Soal 1

```
// MUHAMMAD FAUZAN
// 103112400064
package main

import "fmt"

func selectionSort(angka []int, panjang int) {
    for i := 0; i < panjang-1; i++ {
        indeksTerkecil := i
        for j := i + 1; j < panjang; j++ {
            if angka[j] < angka[indeksTerkecil] {
                indeksTerkecil = j
            }
        }
        angka[i], angka[indeksTerkecil] = angka[indeksTerkecil], angka[i]
    }
}

func cariMedian(angkaTerurut []int, panjang int) int {
    if panjang%2 == 1 {
        return angkaTerurut[panjang/2]
    }
    return (angkaTerurut[panjang/2-1] + angkaTerurut[panjang/2]) / 2
}

func main() {
    var masukan int
    var daftarAngka [1000000]int
    jumlahAngka := 0
    selesai := false

    for !selesai {
        fmt.Scan(&masukan)

        if masukan == -5313 {
            selesai = true
        } else if masukan == 0 {
```

```

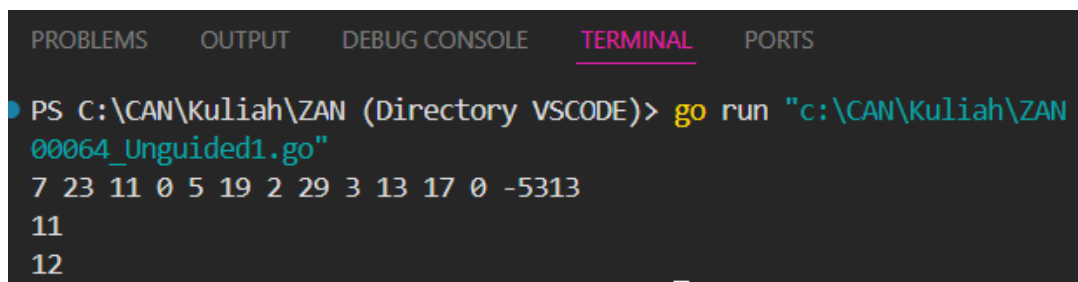
        var salinanAngka [1000000]int
        for i := 0; i < jumlahAngka; i++ {
            salinanAngka[i] = daftarAngka[i]
        }

        selectionSort(salinanAngka[:], jumlahAngka)
        fmt.Println(cariMedian(salinanAngka[:], jumlahAngka))

    } else if masukan > 0 {
        daftarAngka[jumlahAngka] = masukan
        jumlahAngka++
    }
}
}

```

Screenshots Output



```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\CAN\Kuliah\ZAN (Directory VSCODE)> go run "c:\CAN\Kuliah\ZAN
00064_Unguided1.go"
7 23 11 0 5 19 2 29 3 13 17 0 -5313
11
12

```

// Foto hasil dari menjalankan code

Deskripsi: Program ini dirancang untuk menerima masukan berupa bilangan bulat positif secara terus-menerus hingga ditemukan angka -5313 sebagai tanda berhenti. Setiap kali angka 0 dimasukkan, program akan menyalin seluruh data yang telah dikumpulkan, mengurutkannya dengan metode selection sort, lalu menghitung serta menampilkan nilai median dari data tersebut. Tujuan utamanya adalah menyediakan informasi nilai tengah dari data yang terus bertambah selama program berjalan.

Soal 2

```
//MUHAMMAD FAUZAN
//103112400064

package main

import "fmt"

func insertionSort(angka []int) {
    panjang := len(angka)
    for i := 1; i < panjang; i++ {
        kunci := angka[i]
        j := i - 1

        for j >= 0 && angka[j] > kunci {
            angka[j+1] = angka[j]
            j--
        }
        angka[j+1] = kunci
    }
}

func cekJarakTetap(angka []int) (bool, int) {
    if len(angka) < 2 {
        return true, 0
    }

    jarak := angka[1] - angka[0]
    for i := 2; i < len(angka); i++ {
        if angka[i]-angka[i-1] != jarak {
            return false, 0
        }
    }
    return true, jarak
}

func main() {
    var masukan int
    var daftarAngka []int

    for {
        _, err := fmt.Scan(&masukan)
        if err != nil || masukan < 0 {
            continue
        }
        insertionSort(daftarAngka)
        cekJarakTetap(daftarAngka)
    }
}
```

```

        break
    }
    daftarAngka = append(daftarAngka, masukan)
}

insertionSort(daftarAngka)

for _, nilai := range daftarAngka {
    fmt.Print(nilai, " ")
}
fmt.Println()

valid, jarak := cekJarakTetap(daftarAngka)
if valid {
    fmt.Printf("Data berjarak %d\n", jarak)
} else {
    fmt.Println("Data berjarak tidak tetap")
}
}

```

Screenshots Output

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\CAN\Kuliah\ZAN (Directory VSCODE)> go run "c:\CAN\Kuliah\ZAN (Directory VSCODE)\00064_Unguided2.go"
31 13 25 43 1 7 19 37 -5
1 7 13 19 25 31 37 43
Data berjarak 6
PS C:\CAN\Kuliah\ZAN (Directory VSCODE)> go run "c:\CAN\Kuliah\ZAN (Directory VSCODE)\00064_Unguided2.go"
4 40 14 8 26 1 38 2 32 -31
1 2 4 8 14 26 32 38 40
Data berjarak tidak tetap

```

// Foto hasil dari menjalankan code

Deskripsi: Program ini mengumpulkan bilangan bulat non-negatif hingga pengguna memasukkan angka negatif yang menandai akhir input. Setelah seluruh angka dimasukkan, data akan diurutkan menggunakan algoritma insertion sort, kemudian ditampilkan dalam urutan naik. Selanjutnya, program akan memeriksa apakah setiap selisih antara dua elemen berturut-turut selalu sama. Jika ya, program akan menampilkan nilai jaraknya; jika tidak, akan diinformasikan bahwa jarak antar data tidak seragam.

IV. KESIMPULAN

V. REFERENSI

Modul Praktikum Algoritma dan Pemrograman 2. (2025). *Modul 12&13: Pencarian Nilai Ekstrem Pada Himpunan Data*. Fakultas Informatika, Telkom University.