

**LAPORAN PRAKTIKUM  
ALGORITME DAN PEMROGRAMAN 2**

**MODUL 12 DAN 13  
PENGURUTAN DATA**



**Oleh:**

**MUHAMMAD AMIR SALEH**

**2311102233**

**IF - 11 - 06**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA  
FAKULTAS INFORMATIKA  
TELKOM UNIVERSITY PURWOKERTO  
2024**

## **I. Dasar Teori**

Pengurutan data dalam Golang (atau Go) adalah proses menyusun elemen-elemen dalam suatu koleksi data, seperti array atau slice, ke dalam urutan tertentu—baik itu urutan naik (ascending) atau turun (descending). Misalnya, jika kita memiliki daftar angka, pengurutan akan menyusun angka-angka tersebut dari yang terkecil hingga terbesar atau sebaliknya. Golang menyediakan pustaka standar yang memungkinkan kita untuk melakukan pengurutan data dengan menggunakan fungsi `sort`, yang dapat digunakan untuk berbagai jenis data seperti angka, string, dan tipe data lainnya. Fungsi ini memanfaatkan algoritma pengurutan yang efisien, sehingga proses pengurutan bisa dilakukan dengan cepat, bahkan untuk kumpulan data yang besar. Pengurutan ini sangat berguna dalam berbagai aplikasi, seperti pencarian data, penyusunan laporan, atau ketika kita ingin menampilkan data dalam urutan tertentu.

## II. Guided Guided 1

```
package main

import (
    "fmt"
)

// Fungsi untuk mengurutkan array menggunakan Selection Sort
func selectionSort(arr []int, n int) {
    for i := 0; i < n-1; i++ {
        idxMin := i
        for j := i + 1; j < n; j++ {
            // Cari elemen terkecil
            if arr[j] < arr[idxMin] {
                idxMin = j
            }
        }
        // Tukar elemen terkecil dengan elemen di posisi i
        arr[i], arr[idxMin] = arr[idxMin], arr[i]
    }
}

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

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

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

        // Urutkan array dari terkecil ke terbesar
        selectionSort(arr, m)

        // Tampilkan hasil
    }
}
```

```

        fmt.Printf("Nomor rumah terurut untuk daerah %d: ",
daerah)
    for _, num := range arr {
        fmt.Printf("%d ", num)
    }
    fmt.Println()
}
}

```

## Screenshots Output

The screenshot shows a Go IDE with the following components:

- EXPLORER:** Lists files `guided1.go`, `guided2.go`, `unguided1.go`, and `unguided2.go`.
- EDITOR:** Displays the source code for `guided1.go`. It includes a package declaration, imports, a comment in Indonesian, a `selectionSort` function implementing Selection Sort, and a `main` function.
- TERMINAL:** Shows the execution output:
 

```

go run "/Users/namir/Desktop/2311102233_Muhammad Amir Saleh_Modul 12/guided1.go"
namir@MINGW64: ~/Desktop/2311102233_Muhammad Amir Saleh_Modul 12 $ go run "/Users/namir/Desktop/2311102233_Muhammad Amir Saleh_Modul 12/guided1.go"
Masukkan jumlah daerah kerabat (n): 2
Masukkan jumlah nomor rumah kerabat untuk daerah 1: 3
Masukkan 3 nomor rumah kerabat: 66 44 22
Nomor rumah terurut untuk daerah 1: 22 44 66
Masukkan jumlah nomor rumah kerabat untuk daerah 2: 3
Masukkan 3 nomor rumah kerabat: 11 22 33
Nomor rumah terurut untuk daerah 2: 11 22 33

```

## Deskripsi:

Program ini berfungsi untuk mengelola nomor rumah kerabat di beberapa daerah dan mengurutkannya menggunakan algoritma Selection Sort. Pengguna diminta untuk memasukkan jumlah daerah kerabat yang ingin diproses. Untuk setiap daerah, program meminta jumlah nomor rumah dan nomor rumah kerabat tersebut dimasukkan ke dalam array. Setelah itu, program mengurutkan nomor rumah tersebut secara ascending menggunakan algoritma Selection Sort, yang bekerja dengan memilih elemen terkecil pada setiap iterasi dan menukarnya dengan elemen di posisi yang sesuai. Setelah array nomor rumah diurutkan, program menampilkan hasil urutannya untuk setiap daerah, sehingga memudahkan pengguna untuk melihat daftar nomor rumah kerabat yang terurut dengan rapi.

## Guided 2

```
package main

import (
    "fmt"
)

// Fungsi untuk mengurutkan array menggunakan Insertion Sort
func insertionSort(arr []int, n int) {
    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 selisih elemen array tetap
func isConstantDifference(arr []int, n int) (bool, int) {
    if n < 2 {
        return true, 0
    }

    difference := arr[1] - arr[0]
    for i := 1; i < n-1; i++ {
        if arr[i+1]-arr[i] != difference {
            return false, 0
        }
    }
    return true, difference
}

func main() {
    var arr []int
    var num int

    // Input data hingga bilangan negatif ditemukan
    fmt.Println("Masukkan data integer (akhiri dengan bilangan negatif):")
    for {
        fmt.Scan(&num)
```

```

        if num < 0 {
            break
        }
        arr = append(arr, num)
    }

    n := len(arr)

    // Urutkan array menggunakan Insertion Sort
    insertionSort(arr, n)

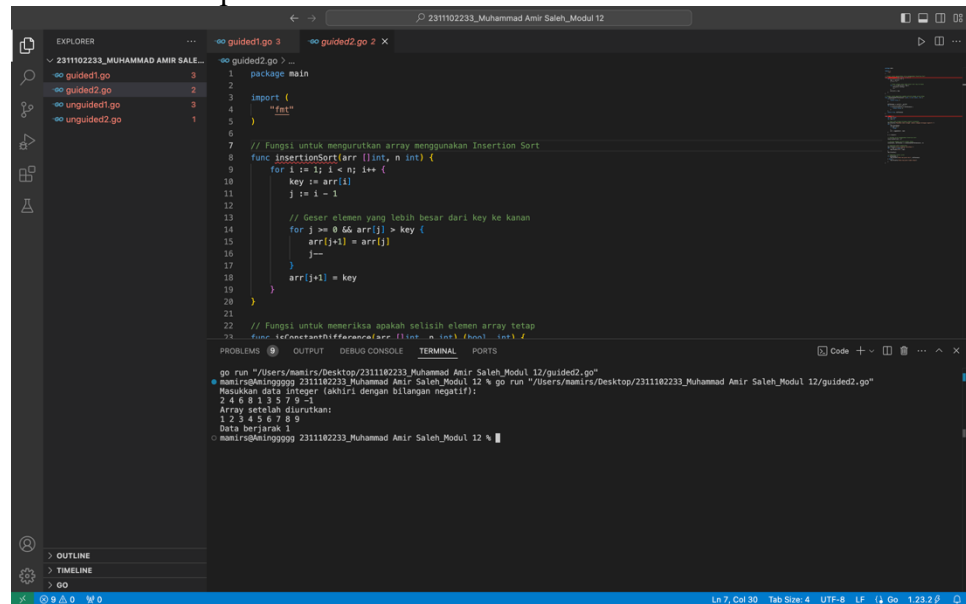
    // Periksa apakah selisih elemen tetap
    isConstant, difference := isConstantDifference(arr, n)

    // Tampilkan hasil pengurutan
    fmt.Println("Array setelah diurutkan:")
    for _, val := range arr {
        fmt.Printf("%d ", val)
    }
    fmt.Println()

    // Tampilkan status jarak
    if isConstant {
        fmt.Printf("Data berjarak %d\n", difference)
    } else {
        fmt.Println("Data berjarak tidak tetap")
    }
}

```

## Screenshots Output



```
1 package main
2
3 import (
4     "fmt"
5 )
6
7 // Fungsi untuk mengurutkan array menggunakan Insertion Sort
8 func insertionSort(arr []int, n int) {
9     for i := 1; i < n; i++ {
10         key := arr[i]
11         j := i - 1
12
13         // Geser elemen yang lebih besar dari key ke kanan
14         for j >= 0 && arr[j] > key {
15             arr[j+1] = arr[j]
16             j--
17         }
18         arr[j+1] = key
19     }
20 }
21
22 // Fungsi untuk memeriksa apakah selisih elemen array tetap
23 func isFunctionDifference(arr []int, n int) bool {
24     // ...
25 }
26
27 func main() {
28     // ...
29 }
30
```

go run "/Users/mamirs/Desktop/2311182233\_Muhammad Amir Saleh\_Modul 12/guided2.go"

masih@Minggggg 2311182233\_Muhammad Amir Saleh\_Modul 12 % go run "/Users/mamirs/Desktop/2311182233\_Muhammad Amir Saleh\_Modul 12/guided2.go"

Masukkan data integer (akhiri dengan bilangan negatif):

2 4 6 8 1 3 5 7 9 -1

Array setelah diurutkan:

1 2 3 4 5 6 7 8 9

Data berjarak 1

## Deskripsi:

Program ini menerima input berupa deretan bilangan bulat yang berakhir dengan bilangan negatif. Input akan disimpan dalam array, lalu array tersebut diurutkan menggunakan algoritma Insertion Sort. Setelah pengurutan, program akan memeriksa apakah selisih antara elemen-elemen array tetap, yaitu apakah ada jarak yang sama antara setiap elemen berturut-turut. Jika selisih antar elemen tetap, program akan menampilkan nilai jarak tersebut; jika tidak, program akan memberi tahu bahwa jaraknya tidak tetap. Program ini efektif untuk mengurutkan data dan memverifikasi konsistensi jarak antar elemen dalam array yang diurutkan.

### III. Unguided Unguided 1

```
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 main() {
    var n int
    fmt.Print("Jumlah daerah: ")
    fmt.Scan(&n)

    for i := 0; i < n; i++ {
        var m int
        fmt.Print("Jumlah kerabat: ")
        fmt.Scan(&m)

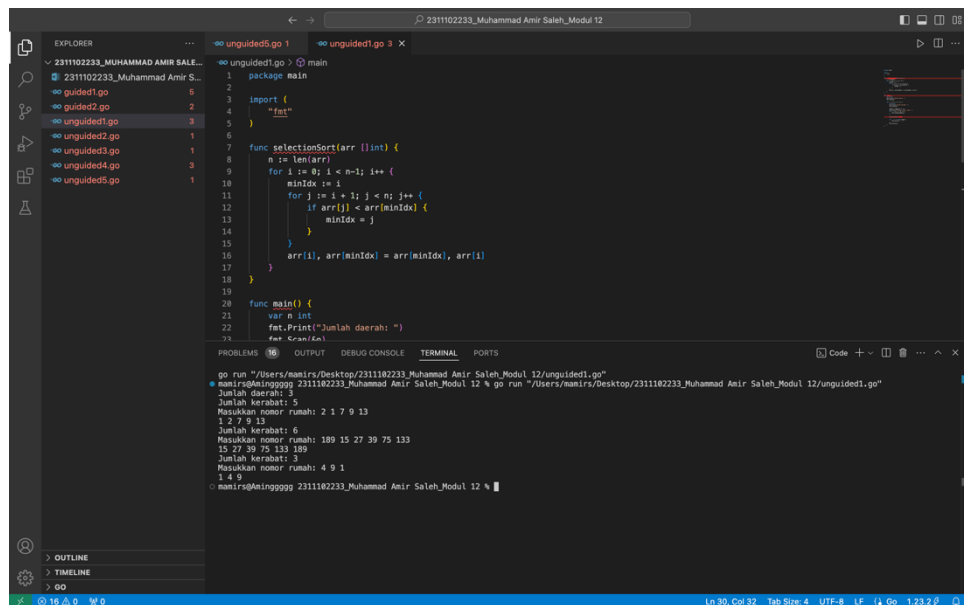
        rumah := make([]int, m)
        fmt.Print("Masukkan nomor rumah: ")
        for j := 0; j < m; j++ {
            fmt.Scan(&rumah[j])
        }

        selectionSort(rumah)

        for _, r := range rumah {
            fmt.Print(r, " ")
        }
        fmt.Println()
    }
}
```



## Screenshots Output



The screenshot shows a Go program in an IDE. The Explorer panel on the left lists files: `231102233_Muhammad Amir S...`, `guided1.go`, `guided2.go`, `unguided1.go`, `unguided2.go`, `unguided3.go`, `unguided4.go`, and `unguided5.go`. The main editor displays the code for `unguided1.go`, which implements a Selection Sort algorithm. The code defines a `selectionSort` function that sorts an array of integers and a `main` function that prompts the user for the number of regions and the house numbers for each region. The output window at the bottom shows the program's execution: it asks for the number of regions (5), then for the house numbers for each region (2, 1, 7, 9, 13), and finally displays the sorted house numbers (1, 2, 7, 9, 13).

```
1 package main
2
3 import (
4     "fmt"
5 )
6
7 func selectionSort(arr []int) {
8     n := len(arr)
9     for i := 0; i < n-1; i++ {
10         minIdx := i
11         for j := i + 1; j < n; j++ {
12             if arr[j] < arr[minIdx] {
13                 minIdx = j
14             }
15         }
16         arr[i], arr[minIdx] = arr[minIdx], arr[i]
17     }
18 }
19
20 func main() {
21     var n int
22     fmt.Print("Jumlah daerah: ")
23     fmt.Scan(&n)
```

```
go run "/Users/mamirs/Desktop/231102233_Muhammad Amir Saleh_Modul 12/unguided1.go"
mamirs@Maringgggg 231102233_Muhammad Amir Saleh_Modul 12 % go run "/Users/mamirs/Desktop/231102233_Muhammad Amir Saleh_Modul 12/unguided1.go"
Jumlah daerah: 5
Masukkan nomor rumah: 2 1 7 9 13
1 2 7 9 13
Jumlah kerabat: 6
Masukkan nomor rumah: 189 15 27 39 75 133
15 27 39 75 133 189
Jumlah kerabat: 3
Masukkan nomor rumah: 4 9 1
1 4 9
```

## Deskripsi:

Program ini meminta input jumlah daerah dan untuk setiap daerah, jumlah kerabat yang memiliki nomor rumah yang akan dimasukkan. Setelah itu, nomor-nomor rumah tersebut diurutkan menggunakan algoritma Selection Sort dari yang terkecil hingga terbesar. Program kemudian menampilkan nomor rumah yang sudah terurut untuk setiap daerah. Dengan menggunakan Selection Sort, program menjamin bahwa nomor rumah yang dimasukkan akan selalu tersusun secara teratur setelah proses pengurutan selesai.

## Unguided 2

```
package main

import (
    "fmt"
)

func selectionSort(arr []int, descending bool) {
    n := len(arr)
    for i := 0; i < n-1; i++ {
        selectedIdx := i
        for j := i + 1; j < n; j++ {
            if (descending && arr[j] > arr[selectedIdx]) ||
                (!descending && arr[j] < arr[selectedIdx]) {
                selectedIdx = j
            }
        }
        arr[i], arr[selectedIdx] = arr[selectedIdx], arr[i]
    }
}

func main() {
    var n int
    fmt.Print("Jumlah daerah: ")
    fmt.Scan(&n)

    for i := 0; i < n; i++ {
        var m int
        fmt.Print("Jumlah kerabat: ")
        fmt.Scan(&m)

        rumah := make([]int, m)
        ganjil := []int{}
        genap := []int{}

        fmt.Print("Masukkan nomor rumah: ")
        for j := 0; j < m; j++ {
            fmt.Scan(&rumah[j])
            if rumah[j]%2 != 0 {
                ganjil = append(ganjil, rumah[j])
            } else {
                genap = append(genap, rumah[j])
            }
        }

        selectionSort(ganjil, false)
    }
}
```

```

        selectionSort(genap, true)

        for _, r := range ganjil {
            fmt.Print(r, " ")
        }
        for _, r := range genap {
            fmt.Print(r, " ")
        }
        fmt.Println()
    }
}

```

## Screenshots Output

The screenshot shows a Go IDE with a file named `unguided2.go`. The code implements a `selectionSort` function that sorts an array in descending order. The `main` function prompts the user for the number of districts (`daerah`) and the number of houses (`rumah`). It then reads the house numbers and sorts them using the `selectionSort` function. The output shows the sorted house numbers for 5 districts: 1 3 5 4 2.

```

1 package main
2
3 import (
4     "fmt"
5 )
6
7 func selectionSort(arr []int, descending bool) {
8     n := len(arr)
9     for i := 0; i < n-1; i++ {
10         selectedIdx := i
11         for j := i+1; j < n; j++ {
12             if (descending && arr[j] > arr[selectedIdx]) || (!descending && arr[j] < arr[selectedIdx]) {
13                 selectedIdx = j
14             }
15         }
16         arr[i], arr[selectedIdx] = arr[selectedIdx], arr[i]
17     }
18 }
19
20 func main() {
21     var n int
22     fmt.Print("Jumlah daerah: ")
23     fmt.Scan(&n)
24 }

```

```

go run "/Users/namir/Desktop/2311102233_Muhammad Amir Saleh_Modul 12/unguided2.go"
Jumlah daerah: 5
Jumlah kerabat: 5
Masukkan nomor rumah: 1 2 3 4 5
1 3 5 4 2

```

## Deskripsi:

Program ini menerima input berupa jumlah daerah dan untuk setiap daerah, jumlah kerabat beserta nomor rumah mereka. Setiap nomor rumah dikelompokkan menjadi dua kategori: bilangan ganjil dan bilangan genap. Nomor rumah ganjil diurutkan dalam urutan menaik (ascending), sedangkan nomor rumah genap diurutkan dalam urutan menurun (descending). Proses pengurutan dilakukan menggunakan algoritma Selection Sort, dengan parameter yang menentukan apakah pengurutan dilakukan secara menaik atau menurun. Setelah pengelompokan dan pengurutan, nomor rumah dari masing-masing kategori dicetak dalam satu baris, dimulai dari nomor ganjil yang sudah terurut diikuti oleh nomor genap yang juga terurut.

### Unguided 3

```
package main

import (
    "fmt"
)

func insertionSort(arr []int) {
    for i := 1; i < len(arr); i++ {
        key := arr[i]
        j := i - 1
        for j >= 0 && arr[j] > key {
            arr[j+1] = arr[j]
            j--
        }
        arr[j+1] = key
    }
}

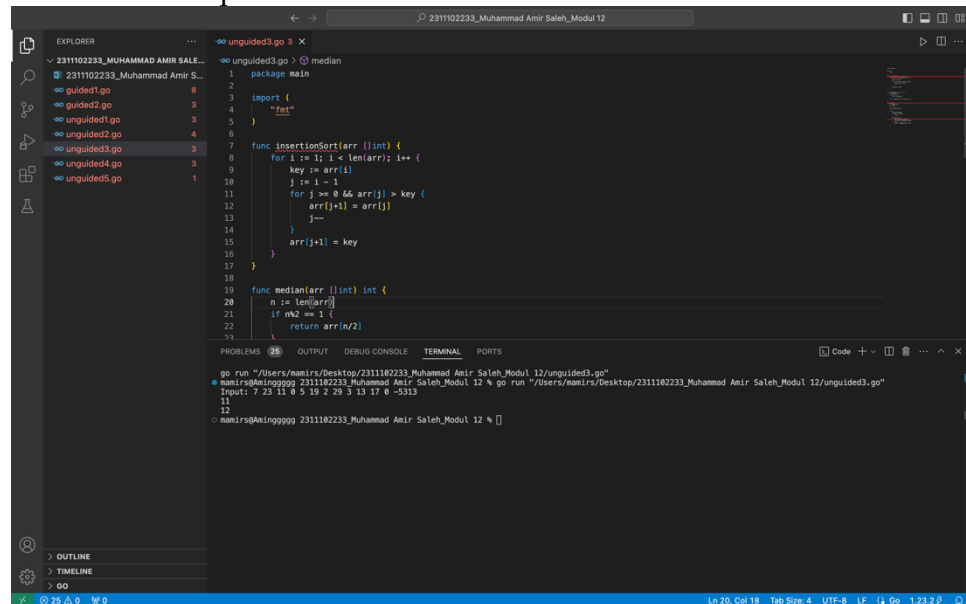
func median(arr []int) int {
    n := len(arr)
    if n%2 == 1 {
        return arr[n/2]
    }
    return (arr[n/2-1] + arr[n/2]) / 2
}

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

    fmt.Print("Input: ")

    for {
        fmt.Scan(&num)
        if num == -5313 {
            break
        }
        if num == 0 {
            insertionSort(data)
            fmt.Println(median(data))
        } else {
            data = append(data, num)
        }
    }
}
```

## Screenshots Output



```
1 package main
2
3 import (
4     "fmt"
5 )
6
7 func InsertionSort(arr []int) {
8     for i := 1; i < len(arr); i++ {
9         key := arr[i]
10        j := i - 1
11        for j >= 0 && arr[j] > key {
12            arr[j+1] = arr[j]
13            j--
14        }
15        arr[j+1] = key
16    }
17 }
18
19 func median(arr []int) int {
20     n := len(arr)
21     if n%2 == 1 {
22         return arr[n/2]
23     }
24 }
```

```
go run "/Users/namir/Desktop/2311102233_Muhammad Amir Saleh_Modul 12/unguided3.go"
namir@MacInggooo 2311102233_Muhammad Amir Saleh_Modul 12 % go run "/Users/namir/Desktop/2311102233_Muhammad Amir Saleh_Modul 12/unguided3.go"
Input: 7 23 11 0 5 19 2 29 3 13 17 0 -5313
11
namir@MacInggooo 2311102233_Muhammad Amir Saleh_Modul 12 %
```

### Deskripsi:

Program ini menerima serangkaian input angka integer yang akan diproses hingga pengguna memasukkan angka -5313, yang menandai akhir dari input. Pengguna dapat memasukkan angka 0 di sela-sela input untuk meminta program menghitung nilai median dari angka-angka yang telah dimasukkan sejauh itu. Saat angka 0 dimasukkan, program akan mengurutkan data menggunakan algoritma Insertion Sort, kemudian menghitung median berdasarkan data yang telah diurutkan. Median dihitung dengan mengambil elemen tengah jika jumlah data ganjil, atau rata-rata dari dua elemen tengah jika jumlah data genap. Semua angka selain 0 dan -5313 akan ditambahkan ke dalam array untuk diproses. Setelah angka -5313 dimasukkan, program akan berhenti.

## Unguided 4

```
package main

import "fmt"

func insertionSort(arr []int) {
    for i := 1; i < len(arr); i++ {
        key := arr[i]
        j := i - 1
        for j >= 0 && arr[j] > key {
            arr[j+1] = arr[j]
            j--
        }
        arr[j+1] = key
    }
}

func cekJarak(arr []int) (bool, int) {
    if len(arr) < 2 {
        return true, 0
    }
    distance := arr[1] - arr[0]
    for i := 2; i < len(arr); i++ {
        if arr[i]-arr[i-1] != distance {
            return false, 0
        }
    }
    return true, distance
}

func main() {
    var arr []int
    var num int

    for {
        fmt.Scan(&num)
        if num < 0 {
            break
        }
        arr = append(arr, num)
    }

    insertionSort(arr)

    for _, val := range arr {
        fmt.Printf("%d ", val)
    }
}
```

```

    }
    fmt.Println()

    isUniform, distance := cekJarak(arr)
    if isUniform {
        fmt.Printf("Data berjarak %d\n", distance)
    } else {
        fmt.Println("Data berjarak tidak tetap")
    }
}

```

### Screenshots Output

The screenshot displays a Go IDE with the following components:

- EXPLORER:** Shows a project structure with files named `unguided1.go` through `unguided5.go`.
- EDITOR:** Displays the source code for `unguided4.go`, which includes an `InsertionSort` function and a `main` function that reads input, sorts it, and checks for uniform spacing using the `cekJarak` function.
- TERMINAL:** Shows the command `go run "/Users/manirs/Desktop/2311182233_Muhammad Amir Saleh_Modul 12/unguided4.go"` and its output:
 

```

2 1 4 5 6 1 7 8 9
-1
0 1 2 3 4 5 6 7 8 9
Data berjarak 1

```

### Deskripsi:

Program ini menerima input berupa angka-angka yang dimasukkan oleh pengguna hingga ditemukan angka negatif, yang menandakan akhir input. Angka-angka tersebut kemudian disimpan dalam sebuah array dan diurutkan menggunakan algoritma Insertion Sort. Setelah array terurut, program memeriksa apakah jarak antara elemen-elemen dalam array konsisten dengan membandingkan selisih antara elemen yang berurutan. Jika jarak antar elemen tetap, program akan menampilkan jarak tersebut; jika tidak, program akan memberi tahu bahwa jaraknya tidak tetap. Hasil akhirnya adalah array yang terurut dan informasi mengenai jarak antar elemen dalam array tersebut.

## Unguided 5

```
package main

import (
    "fmt"
)

const nMax int = 7919

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

type DaftarBuku [nMax]Buku

func DaftarkanBuku(pustaka *DaftarBuku, n int) {
    for i := 0; i < n; i++ {
        fmt.Printf("Data buku ke-%d:\n", i+1)
        fmt.Print("ID: ")
        fmt.Scan(&pustaka[i].id)
        fmt.Print("Judul: ")
        fmt.Scan(&pustaka[i].judul)
        fmt.Print("Penulis: ")
        fmt.Scan(&pustaka[i].penulis)
        fmt.Print("Penerbit: ")
        fmt.Scan(&pustaka[i].penerbit)
        fmt.Print("Eksemplar: ")
        fmt.Scan(&pustaka[i].eksemplar)
        fmt.Print("Tahun: ")
        fmt.Scan(&pustaka[i].tahun)
        fmt.Print("Rating: ")
        fmt.Scan(&pustaka[i].rating)
    }
}

func CetakTerfavorit(pustaka DaftarBuku, n int) {
    if n == 0 {
        fmt.Println("Tidak ada buku.")
        return
    }
    maxIndex := 0
    for i := 1; i < n; i++ {
        if pustaka[i].rating > pustaka[maxIndex].rating {
            maxIndex = i
        }
    }
}
```



```

    }
    fmt.Println("Buku terfavorit:")
    fmt.Printf("Judul: %s, Penulis: %s, Penerbit: %s, Tahun:
%d\n",
        pustaka[maxIndex].judul, pustaka[maxIndex].penulis,
        pustaka[maxIndex].penerbit, pustaka[maxIndex].tahun)
}

func UrutBuku(pustaka *DaftarBuku, n int) {
    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 Cetak5Terbaru(pustaka DaftarBuku, n int) {
    fmt.Println("5 Buku dengan rating tertinggi:")
    count := 5
    if n < 5 {
        count = n
    }
    for i := 0; i < count; i++ {
        fmt.Printf("%d. %s\n", i+1, pustaka[i].judul)
    }
}

func CariBuku(pustaka DaftarBuku, n, r int) {
    left, right := 0, n-1
    for left <= right {
        mid := (left + right) / 2
        if pustaka[mid].rating == r {
            fmt.Println("Buku ditemukan:")
            fmt.Printf("Judul: %s, Penulis: %s, Penerbit: %s,
Tahun: %d, Eksemplar: %d, Rating: %d\n",
                pustaka[mid].judul, pustaka[mid].penulis,
                pustaka[mid].penerbit,
                pustaka[mid].tahun, pustaka[mid].eksemplar,
                pustaka[mid].rating)
            return
        } else if pustaka[mid].rating > r {
            left = mid + 1

```

```

    } else {
        right = mid - 1
    }
}
fmt.Println("Tidak ada buku dengan rating seperti itu.")
}

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

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

    DaftarkanBuku(&pustaka, n)

    CetakTerfavorit(pustaka, n)

    UrutBuku(&pustaka, n)

    Cetak5Terbaru(pustaka, n)

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

```

## Screenshots Output

The screenshot shows a Go IDE interface. The top bar displays the file path: 2311102233\_Muhammad Amir Saleh\_Modul 12. The left sidebar contains a file explorer with a tree view showing a project structure for '2311102233\_Muhammad Amir Saleh'. The main editor area displays the source code for 'unguided5.go', which includes a package declaration, imports, constants, a struct definition for 'Buku', a slice definition for 'DaftarBuku', and a function 'DaftarBuku' that iterates over the slice and prints book details. The bottom panel shows a terminal window with the output of the program, displaying the details of two books: 'Data buku ke-1' and 'Data buku ke-2'.

#### Deskripsi:

Program ini mengelola data buku dengan beberapa fitur untuk mengolah dan mencari informasi tentang buku yang terdaftar. Program dimulai dengan mendefinisikan sebuah array DaftarBuku yang berisi data buku, termasuk ID, judul, penulis, penerbit, jumlah eksemplar, tahun terbit, dan rating. Pengguna diminta untuk memasukkan data buku, yang kemudian disimpan dalam array tersebut. Fungsi DaftarkanBuku digunakan untuk memasukkan data buku ke dalam array. Setelah itu, program menampilkan buku dengan rating tertinggi menggunakan fungsi CetakTerfavorit. Kemudian, buku-buku diurutkan berdasarkan rating menggunakan algoritma pengurutan UrutBuku untuk menampilkan daftar buku dengan rating tertinggi. Fungsi Cetak5Terbaru digunakan untuk menampilkan lima buku dengan rating tertinggi atau sebanyak jumlah buku yang ada jika kurang dari lima. Terakhir, pengguna dapat mencari buku berdasarkan rating tertentu dengan memasukkan rating melalui fungsi CariBuku, yang akan melakukan pencarian dengan metode binary search untuk menemukan buku yang memiliki rating tersebut.