

**LAPORAN PRAKTIKUM
ALGORITMA PROGRAM 2
MODUL 12
PENGURUTAN DATA**



Oleh:

ADITHANA DHARMA PUTRA

2311102207

IF – 11- 02

**S1 TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
2024**

1. DASAR TEORI

12.1 Ide Algoritma Selection Sort

Pengurutan secara seleksi ini idenya adalah mencari nilai ekstrim pada sekumpulan data, kemudian meletakkan pada posisi yang seharusnya. Pada penjelasan berikut ini data akan diurut membesar (*ascending*), dan data dengan indeks kecil ada di "kiri" dan indeks besar ada di "kanan".

- 1) Cari nilai terkecil di dalam rentang data tersisa
- 2) Pindahkan/tukar tempat dengan data yang berada pada posisi paling kiri pada rentang data tersisa tersebut.
- 3) Ulangi proses ini sampai tersisa hanya satu data saja.

Algoritma ini dikenal juga dengan nama Selection Sort, yang mana pada algoritma ini melibatkan dua proses yaitu pencarian indeks nilai ekstrim dan proses pertukaran dua nilai atau *swap*.

	Notasi Algoritma	Notasi dalam bahasa Go
1	$i \leftarrow 1$	$i = 1$
2	while $i \leq n-1$ do	for $i \leq n-1$ {
3	$idx_min \leftarrow i - 1$	$idx_min = i - 1$
4	$j \leftarrow i$	$j = i$
5	while $j < n$ do	for $j < n$ {
6	if $a[idx_min] > a[j]$ then	if $a[idx_min] > a[j]$ {
7	$idx_min \leftarrow j$	$idx_min = j$
8	endif	}
9	$j \leftarrow j + 1$	$j = j + 1$
10	endwhile	}
11	$t \leftarrow a[idx_min]$	$t = a[idx_min]$
12	$a[idx_min] \leftarrow a[i-1]$	$a[idx_min] = a[i-1]$
13	$a[i-1] \leftarrow t$	$a[i-1] = t$
14	$i \leftarrow i + 1$	$i = i + 1$
15	endwhile	}

12.2 Algoritma Selection Sort

Adapun algoritma *selection* sort pada untuk mengurutkan array bertipe data bilangan bulat secara membesar atau ascending adalah sebagai berikut ini!

```
..
..
5  type arrInt [4321]int
..
..
15 func selectionSort1(T *arrInt, n int){
16 /* I.S. terdefinisi array T yang berisi n bilangan bulat
17    F.S. array T terurut secara ascending atau membesar dengan SELECTION SORT */
18    var t, i, j, idx_min int
19
```

```

20     i = 1
21     for i <= n-1 {
22         idx_min = i - 1
23         j = i
24         for j < n {
25             if T[idx_min] > T[j] {
26                 idx_min = j
27             }
28             j = j + 1
29         }
30         t = T[idx_min]
31         T[idx_min] = T[i-1]
32         T[i-1] = t
33         i = i + 1
34     }
35 }

```

Sama halnya apabila array yang akan diurutkan adalah bertipe data struct, maka tambahkan field pada saat proses perbandingan nilai ekstrim, kemudian tipe data dari variabel t sama dengan struct dari arraynya.

```

..     ...
5     type mahasiswa struct {
..         nama, nim, kelas, jurusan string
..         ipk float64
..     }
..     type arrMhs [2023]mahasiswa
..     ...
15 func selectionSort2(T * arrMhs, n int){
16     /* I.S. terdefinisi array T yang berisi n data mahasiswa
17        F.S. array T terurut secara ascending atau membesar berdasarkan ipk dengan
18        menggunakan algoritma SELECTION SORT */
19     var i, j, idx_min int
20     var t mahasiswa
21     i = 1
22     for i <= n-1 {
23         idx_min = i - 1
24         j = i
25         for j < n {
26             if T[idx_min].ipk > T[j].ipk {
27                 idx_min = j
28             }
29             j = j + 1
30         }
31         t = T[idx_min]
32         T[idx_min] = T[i-1]
33         T[i-1] = t
34         i = i + 1
35     }
36 }

```

12.4 Ide Algoritma Insertion Sort

Pengurutan secara insertion ini idenya adalah menyisipkan suatu nilai pada posisi yang seharusnya. Berbeda dengan pengurutan seleksi, yang mana pada pengurutan ini tidak dilakukan pencarian nilai ekstrim terlebih dahulu, cukup memilih suatu nilai tertentu kemudian mencari posisinya secara *sequential search*. Pada penjelasan berikut ini data akan diurut mengecil (*descending*), dan data dengan indeks kecil ada di "kiri" dan indeks besar ada di "kanan".

- 1) Untuk satu data yang belum terurut dan sejumlah data yang sudah diurutkan:

Geser data yang sudah terurut tersebut (ke kanan), sehingga ada satu ruang kosong untuk memasukkan data yang belum terurut ke dalam data yang sudah terurut dan tetap menjaga keterurutan.

- 2) Ulangi proses tersebut untuk setiap data yang belum terurut terhadap rangkaian data yang sudah terurut.

Algoritma ini dikenal juga dengan nama *Insertion Sort*, yang mana pada algoritma ini melibatkan dua proses yaitu pencarian sekuensial dan penyisipan.

	Notasi Algoritma	Notasi dalam bahasa Go
1	$i \leftarrow 1$	$i = 1$
2	while $i \leq n-1$ do	for $i \leq n-1$ {
3	$j \leftarrow i$	$j = i$
4	$temp \leftarrow a[j]$	$temp = a[j]$
5	while $j > 0$ and $temp > a[j-1]$ do	for $j > 0$ && $temp > a[j-1]$ {
6	$a[j] \leftarrow a[j-1]$	$a[j] = a[j-1]$
7	$j \leftarrow j - 1$	$j = j - 1$
8	endwhile	}
9	$a[j] \leftarrow temp$	$a[j] = temp$
10	$i \leftarrow i + 1$	$i = i + 1$
11	endwhile	}

12.5 Algoritma Insertion Sort

Adapun algoritma *insertion sort* pada untuk mengurutkan array bertipe data bilangan bulat secara mengecil atau *descending* adalah sebagai berikut ini!

```
..    ...
5    type arrInt [4321]int
..    ...
15   func insertionSort1(T *arrInt, n int){
16   /* I.S. terdefinisi array T yang berisi n bilangan bulat
17      F.S. array T terurut secara mengecil atau descending dengan INSERTION SORT*/
18      var temp, i, j int
19      i = 1
20      for i <= n-1 {
21          j = i
22          temp = T[j]
23          for j > 0 && temp > T[j-1] {
24              T[j] = T[j-1]
25              j = j - 1
26          }
27          T[j] = temp
28          i = i + 1
29      }
30  }
```

Sama halnya apabila array yang akan diurutkan adalah bertipe data struct, maka tambahkan field pada saat proses perbandingan dalam pencarian posisi, kemudian tipe data dari variabel *temp* sama dengan struct dari arraynya.

```
..    ...
5    type mahasiswa struct {
..      nama, nim, kelas, jurusan string
..      ipk float64
..    }
..    type arrMhs [2023]mahasiswa
..    ...
15   func insertionSort2(T * arrMhs, n int){
16   /* I.S. terdefinisi array T yang berisi n data mahasiswa
17      F.S. array T terurut secara mengecil atau descending berdasarkan nama dengan
18      menggunakan algoritma INSERTION SORT */
19      var temp i, j int
20      var temp mahasiswa
21      i = 1
22      for i <= n-1 {
23          j = i
24          temp = T[j]
25          for j > 0 && temp.nama > T[j-1].nama {
26              T[j] = T[j-1]
27              j = j - 1
28          }
29          T[j] = temp
30          i = i + 1
}
```

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
            }

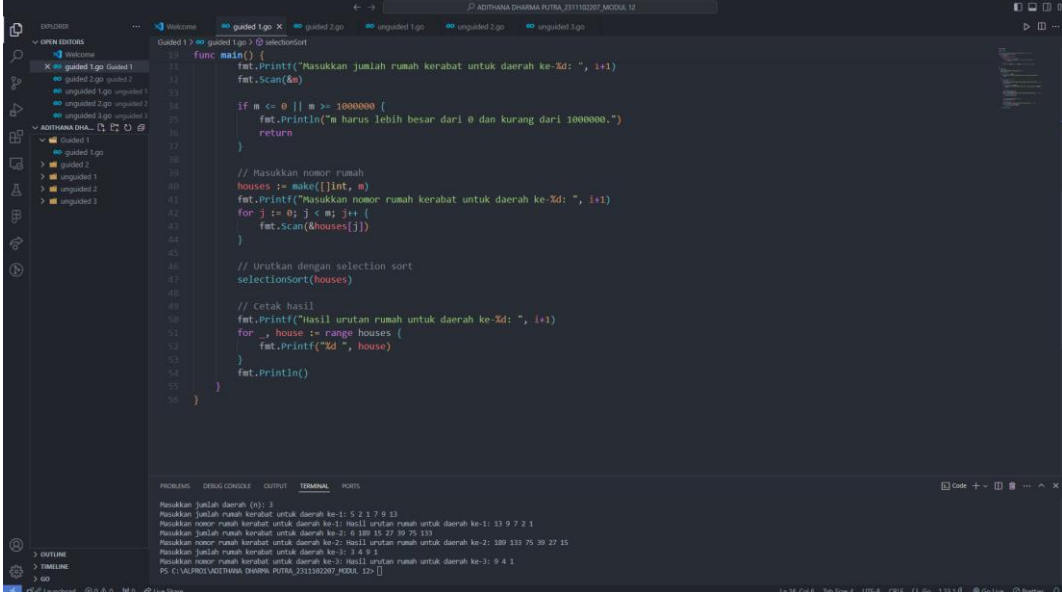
            // Masukkan nomor rumah
            houses := make([]int, m)
            fmt.Printf("Masukkan nomor rumah
kerabat untuk daerah ke-%d: ", i+1)
            for j := 0; j < m; j++ {
                fmt.Scan(&houses[j])
            }

            // 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



```
func main() {
    fmt.Println("Masukkan jumlah rumah kerabat untuk daerah ke-1:", i+1)
    fmt.Scan(&n)

    if n <= 0 || n >= 1000000 {
        fmt.Println("n harus lebih besar dari 0 dan kurang dari 1000000.")
        return
    }

    // Masukkan nomor rumah
    houses := make([]int, n)
    fmt.Println("Masukkan nomor rumah kerabat untuk daerah ke-1:", i+1)
    for j := 0; j < n; j++ {
        fmt.Scan(&houses[j])
    }

    // Urutkan dengan selection sort
    selectionSort(houses)

    // Cetak hasil
    fmt.Println("Hasil urutan rumah untuk daerah ke-1:", i+1)
    for _, house := range houses {
        fmt.Printf("%d ", house)
    }
    fmt.Println()
}
```

PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL PORTS

Masukkan jumlah daerah (n): 3
Masukkan jumlah rumah kerabat untuk daerah ke-1: 5 2 1 7 9 13
Masukkan nomor rumah kerabat untuk daerah ke-1: Hasil urutan rumah untuk daerah ke-1: 13 9 7 2 1
Masukkan jumlah rumah kerabat untuk daerah ke-2: 6 10 15 27 39 75 133
Masukkan nomor rumah kerabat untuk daerah ke-2: Hasil urutan rumah untuk daerah ke-2: 10 15 27 39 75 133
Masukkan jumlah rumah kerabat untuk daerah ke-3: 3 4 8 1
Masukkan nomor rumah kerabat untuk daerah ke-3: Hasil urutan rumah untuk daerah ke-3: 4 1
PS C:\VALPRO2\ADITHIANA Dharma PUTRA_231110207_MDC04_12>

Deskripsi:

Kode ini adalah program yang meminta pengguna untuk memasukkan jumlah daerah dan jumlah rumah kerabat di setiap daerah, kemudian mengurutkan nomor rumah tersebut menggunakan algoritma selection sort. Pertama, program meminta pengguna untuk memasukkan jumlah daerah. Untuk setiap daerah, program meminta pengguna untuk memasukkan jumlah rumah dan nomor rumah kerabat di daerah tersebut. Setelah itu, program mengurutkan nomor rumah menggunakan selection sort dan menampilkan hasil urutan nomor rumah untuk setiap daerah.

2. 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

```

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])))
    }
}

func main() {
    // Contoh penggunaan
    arr := []int{11, 25, 43, 1, 7, 19, 17, -1}
    InsertionSort(arr)
    fmt.Println("Hasil pengurutan: ", arr)
    isConsistent, diff := isDataConsistentlySpaced(arr)
    if isConsistent {
        fmt.Printf("Data berjarak %d\n", diff)
    } else {
        fmt.Println("Data berjarak tidak tetap")
    }
}

```

Deskripsi:

program di atas digunakan untuk mengurutkan array menggunakan insertion sort dan memeriksa apakah data dalam array tersebut berjarak tetap. Program ini meminta pengguna untuk memasukkan data integer yang diakhiri dengan bilangan negatif. Setelah data dimasukkan, program mengurutkan data menggunakan insertion sort. Kemudian, program memeriksa apakah selisih antara elemen-elemen dalam array yang sudah diurutkan adalah konstan. Jika selisihnya konstan, program mencetak bahwa data berjarak tetap beserta nilai selisihnya. Jika tidak, program mencetak bahwa data tidak berjarak tetap.

II. UNGUIDED

1. Unguided 1

Source Code

```
package main
import "fmt"

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] {
                maxIdx = j
            }
        }
        arr[i], arr[maxIdx] = arr[maxIdx], arr[i]
    }
}

func main() {
    var n int
    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.Scan(&m)

        if m <= 0 || m >= 1000000 {
```

```

        fmt.Println("harus lebih besar dari 0
dan kurang dari 1000000.")
        return
    }

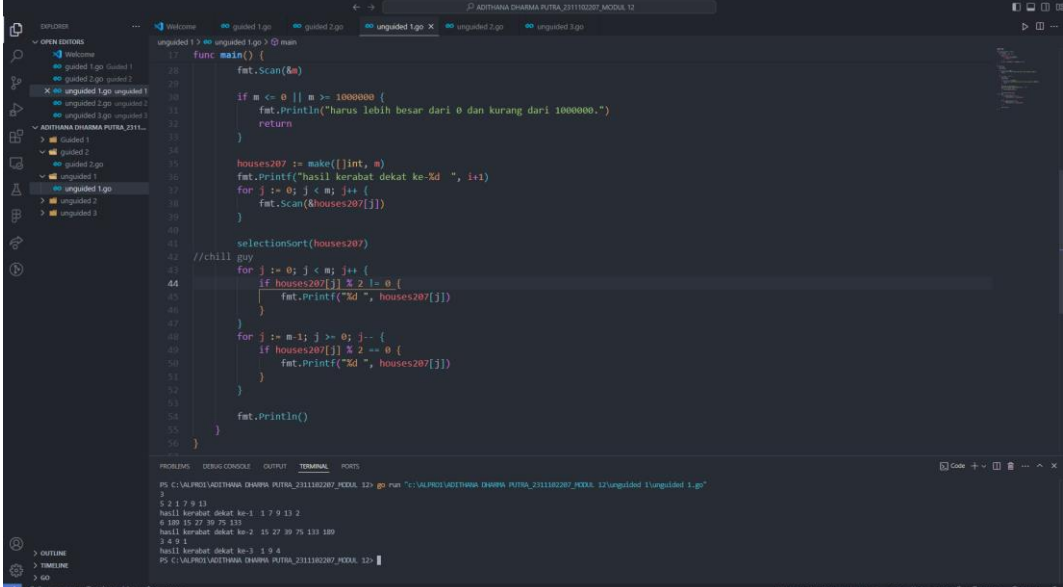
    houses207 := make([]int, m)
    fmt.Printf("hasil kerabat dekat ke-%d ",
i+1)
    for j := 0; j < m; j++ {
        fmt.Scan(&houses207[j])
    }

    selectionSort(houses207)
//chill guy
    for j := 0; j < m; j++ {
        if houses207[j] % 2 != 0 {
            fmt.Printf("%d ", houses207[j])
        }
    }
    for j := m-1; j >= 0; j-- {
        if houses207[j] % 2 == 0 {
            fmt.Printf("%d ", houses207[j])
        }
    }

    fmt.Println()
}
}

```

Screenshot



```
func main() {  
    fmt.Scan(&n)  
  
    if n <= 0 || n >= 1000000 {  
        fmt.Println("harus lebih besar dari 0 dan kurang dari 1000000.")  
        return  
    }  
  
    houses207 := make([]int, n)  
    fmt.Printf("hasil kerabat dekat ke-%d ", i+1)  
    for j := 0; j < n; j++ {  
        fmt.Scan(&houses207[j])  
    }  
  
    selectionSort(houses207)  
  
    //chill guy  
    for j := 0; j < n; j++ {  
        if houses207[j] % 2 != 0 {  
            fmt.Printf("%d ", houses207[j])  
        }  
    }  
    for j := n-1; j >= 0; j-- {  
        if houses207[j] % 2 == 0 {  
            fmt.Printf("%d ", houses207[j])  
        }  
    }  
    fmt.Println()  
}
```

PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL PORTS

```
PS C:\VALPROJ\ADITHIANA DHARMA PUTRA_2311180207_PEXXA_12> go run "C:\VALPROJ\ADITHIANA DHARMA PUTRA_2311180207_PEXXA_12\unguided 1\unguided 1.go"  
5  
5 2 1 7 9 11  
hasil kerabat dekat ke-1: 1 7 9 11 2  
6 100 15 27 39 75 131  
hasil kerabat dekat ke-2: 15 27 39 75 131 100  
3 4 9 1  
hasil kerabat dekat ke-3: 1 9 4
```

Deskripsi

Pada program diatas saya hanya melakukan sedikit perubahan dari guided 1 dimana saya menambahkan iterasi untuk mencetak dari indeks belakang untuk genap dan indeks depan untuk ganjil. Lebih rinci pada Fungsi selectionSort program Mengurutkan array arr menggunakan algoritma selection sort. Kemudian Mencari elemen terkecil dalam subarray yang belum diurutkan dan menukarnya dengan elemen pertama dari subarray tersebut.

2. Unguided 2

Source Code

```
package main

import "fmt"

func selectionSort(arr207 []int) {
    n := len(arr207)
    for i := 0; i < n-1; i++ {
        maxIdx := i
        for j := i + 1; j < n; j++ {
            if arr207[j] < arr207[maxIdx] {
                maxIdx = j
            }
        }
        arr207[i], arr207[maxIdx] =
arr207[maxIdx], arr207[i]
    }
}

func main() {
    var arr207 []int
    var slice []int
    var n int

    var median int
    for {
        fmt.Scan(&n)
        arr207 = append(arr207, n)
        if n == -5313 {
            break
        }
    }
    panjang := len(arr207)

    for i := 0; i < panjang; i++ {
```

```

        if arr207[i] == 0 {
            selectionSort(slice)
            if len(slice)%2 == 0 {

                median = (slice[(len(slice)/2)-1]
+ slice[len(slice)/2]) / 2

            } else {
                median = slice[len(slice)/2]
                fmt.Print()
            }
            fmt.Println(median)
        } else {
            slice = append(slice, arr207[i])
        }
    }
}

```

Screenshot

The screenshot shows a Go IDE with a project named 'ADITHYANA DHARMA PUTRA_231102027_AKOSIA_12'. The main.go file contains the following code:

```

16 func main() {
17     var arr207 []int
18     var slice []int
19     var n int
20
21     var median int
22     for {
23         fmt.Scan(&n)
24         arr207 = append(arr207, n)
25         if n == -5113 {
26             break
27         }
28     }
29     panjang := len(arr207)
30
31     for i := 0; i < panjang; i++ {
32         if arr207[i] == 0 {
33             selectionSort(slice)
34             if len(slice)%2 == 0 {
35
36                 median = (slice[(len(slice)/2)-1] + slice[len(slice)/2]) / 2
37
38             } else {
39                 median = slice[len(slice)/2]
40                 fmt.Print()
41             }
42             fmt.Println(median)
43         } else {
44             slice = append(slice, arr207[i])
45         }
46     }
47 }

```

The terminal output shows the execution of the program, displaying the array elements and the calculated median values.

```

PS C:\VA\PROJ\ADITHYANA DHARMA PUTRA_231102027_AKOSIA_12> go run ".\main.go"
7 23 11 0 5 10 2 20 9 13 17 0 -5113
11
12
PS C:\VA\PROJ\ADITHYANA DHARMA PUTRA_231102027_AKOSIA_12>

```


Deskripsi

Ini adalah program yang mengurutkan array menggunakan selection sort dan menghitung median dari subarray yang dipisahkan oleh nilai 0. Program ini pertama-tama meminta pengguna untuk memasukkan serangkaian angka yang akan disimpan dalam array arr207. Input berakhir ketika pengguna memasukkan angka -5313. Setelah itu, program memproses array arr207 dan setiap kali menemukan nilai 0, ia mengurutkan subarray slice yang berisi angka-angka sebelum 0 tersebut. Jika panjang slice genap, median dihitung sebagai rata-rata dari dua elemen tengah. Jika panjang slice ganjil, median adalah elemen tengah dari slice. Median kemudian dicetak. Jika elemen yang diproses bukan 0, elemen tersebut ditambahkan ke slice. Program ini terus mengulangi proses ini untuk setiap elemen dalam arr207.

3. Unguided 3

Source Code

```
package main
import "fmt"

const nmax = 7919

type Buku struct {
    ID, Judul, Penulis, Penerbit string
    Eksemplar, Tahun, Ranting      int
}
type DaftarBuku [nmax]Buku

func insertionSort(Pustaka207 *DaftarBuku, n int)
{
    for i := 1; i < n; i++ {
        key := Pustaka207[i]
        j := i - 1
```

```

        for j >= 0 && Pustaka207[j].Ranting <
key.Ranting {
            Pustaka207[j+1] = Pustaka207[j]
            j--
        }
        Pustaka207[j+1] = key
    }
}

func CariFavorit(Pustaka207 DaftarBuku, n int) {
    max := Pustaka207[0].Ranting
    favorit := 0
    for i := 0; i < n; i++ {
        if Pustaka207[i].Ranting > max {
            max = Pustaka207[i].Ranting
            favorit = i
        }
    }
    fmt.Printf("Buku Terfavorit Adalah : %s %s %s
%s %v \n", Pustaka207[favorit].ID,
Pustaka207[favorit].Judul,
Pustaka207[favorit].Penulis,
Pustaka207[favorit].Penerbit,
Pustaka207[favorit].Tahun)
}

func Rating(Pustaka207 DaftarBuku, n int) {
    insertionSort(&Pustaka207, n)
    fmt.Print("Lima Rating Tertinggi : ")
    for i := 0; i < n; i++ {
        fmt.Print(Pustaka207[i].Judul, " ")
    }
}

```

```

        fmt.Println()
    }

    func BinarySearch(Pustaka207 DaftarBuku, n, target
    int) int {
        low, high := 0, n-1

        for low <= high {
            mid := (low + high) / 2

            if Pustaka207[mid].Ranting == target {
                return mid
            } else if Pustaka207[mid].Ranting < target
        {
            low = mid + 1
        } else {
            high = mid - 1
        }
    }

    return -1
}

func main() {
    var Buku DaftarBuku
    var n, Cari int
    fmt.Print("Masukan Banyak Buku : ")
    fmt.Scan(&n)
    fmt.Println("Masukan (ID, Judul, Penulis,
    Penerbit, Eksemplar, Tahun, Rating)")
    for i := 0; i < n; i++ {
        fmt.Print("Masukan : ")
        fmt.Scan(&Buku[i].ID, &Buku[i].Judul,
        &Buku[i].Penulis, &Buku[i].Penerbit,

```

```

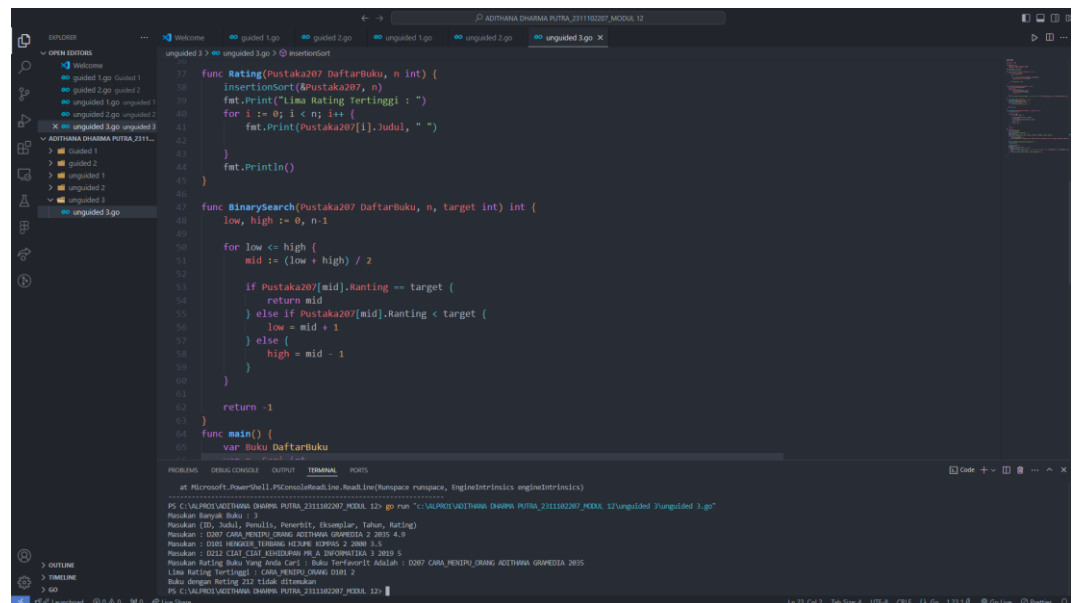
        &Buku[i].Eksemplar, &Buku[i].Tahun,
        &Buku[i].Ranting)
    }

    fmt.Print("Masukan Rating Buku Yang Anda Cari
: ")
    fmt.Scan(&Cari)

    CariFavorit(Buku, n)
    Rating(Buku, n)
    Temukan := BinarySearch(Buku, n, Cari)
    if Temukan != -1 {
        fmt.Printf("Buku dengan Rating %v : %s %s
%s %s %v %v %v\n", Cari, Buku[Temukan].ID,
Buku[Temukan].Judul, Buku[Temukan].Penulis,
Buku[Temukan].Penerbit, Buku[Temukan].Eksemplar,
Buku[Temukan].Tahun, Buku[Temukan].Ranting)
    } else {
        fmt.Printf("Buku dengan Reting %v tidak
ditemukan", Cari)
    }
}

```

Screenshot



```
func Rating(Pustaka207 DaftarBuku, n int) {
    insertionSort(&Pustaka207, n)
    fmt.Println("Lima Rating tertinggi : ")
    for i := 0; i < n; i++ {
        fmt.Print(Pustaka207[i].Judul, " ")
    }
    fmt.Println()
}

func BinarySearch(Pustaka207 DaftarBuku, n, target int) int {
    low, high := 0, n-1

    for low <= high {
        mid := (low + high) / 2
        if Pustaka207[mid].Rating == target {
            return mid
        } else if Pustaka207[mid].Rating < target {
            low = mid + 1
        } else {
            high = mid - 1
        }
    }

    return -1
}

func main() {
    var Buku DaftarBuku
    // ... (rest of the code) ...
}
```

Terminal Output:

```
PS C:\VALPROJ\ADITHIANA_DWIYANG_PUTRA_2311180207_MK004_12> go run "C:\VALPROJ\ADITHIANA_DWIYANG_PUTRA_2311180207_MK004_12\unguided\unguided_3.go"
Masukan Banyak Buku : 5
Masukan (ID, Judul, Penulis, Penerbit, Eksemplar, Tahun, Rating)
Masukan : 0007 CARA MENYIMPAN ADITHIANA DWIYANG 2 2015 4,9
Masukan : 0001 INOVASI TEKNOLOGI KIRYUS 2 2000 3,5
Masukan : 0212 CAT CAT KENDAHAN PA_A INFORMATIKA 3 2019 5
Masukan Rating Buku Yang Anda Cari : Buku Terfavorit Adalah : 0007 CARA MENYIMPAN ADITHIANA DWIYANG 2015
Lima Rating tertinggi : CARA MENYIMPAN ADITHIANA DWIYANG 2015
Buku dengan Rating 022 tidak ditemukan
PS C:\VALPROJ\ADITHIANA_DWIYANG_PUTRA_2311180207_MK004_12>
```

Deskripsi

Ini adalah program untuk mengelola daftar buku dalam sebuah perpustakaan. Program ini menggunakan struktur data Buku untuk menyimpan informasi tentang buku, termasuk ID, judul, penulis, penerbit, jumlah eksemplar, tahun, dan rating. Program ini memiliki beberapa fungsi utama: `insertionSort` untuk mengurutkan buku berdasarkan rating secara menurun, `CariFavorit` untuk menemukan buku dengan rating tertinggi, `Rating` untuk mencetak lima buku dengan rating tertinggi, dan `BinarySearch` untuk mencari buku berdasarkan rating menggunakan algoritma pencarian biner. Dalam fungsi `main`, program meminta pengguna untuk memasukkan jumlah buku dan detail setiap buku, kemudian meminta rating buku yang ingin dicari. Program kemudian menampilkan buku dengan rating tertinggi, lima buku dengan rating tertinggi, dan hasil pencarian buku berdasarkan rating yang dimasukkan pengguna. Jika buku dengan rating yang dicari ditemukan, program mencetak detail buku tersebut; jika tidak, program mencetak pesan bahwa buku dengan rating tersebut tidak ditemukan.