LAPORAN PRAKTIKUM
ALGORITMA PEMROGRAMAN 2

MODUL 12

SORTING

Oleh:

PANDIA ARYA BRATA

2311102076

IF – 11 - 02

S1 TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2024

I. 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 ← 1	i = 1
2	while i <= n-1 do	for i <= n-1 {
3	idx_min ← i - 1	idx_min = i - 1
4	j ← 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 ← j	idx_min = j
8	endif	}
9	j ← j + 1	j = j + 1
10	endwhile	}
11	t ← a[idx_min]	t = a[idx_min]
12	a[idx_min] ← a[i-1]	a[idx_min] = a[i-1]
13	a[i-1] ← t	a[i-1] = t
14	i ← 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 asceding 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 asceding 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 ← 1`	`i = 1`
2	`while i <= n-1 do`	`for i <= n-1 {`
3	`    j ← i`	`    j = i`
4	`    temp ← 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] ← a[j-1]`	`        a[j] = a[j-1]`
7	`        j ← j - 1`	`        j = j - 1`
8	`    endwhile`	`    }`
9	`    a[j] ← temp`	`    a[j] = temp`
10	`    i ← 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
```

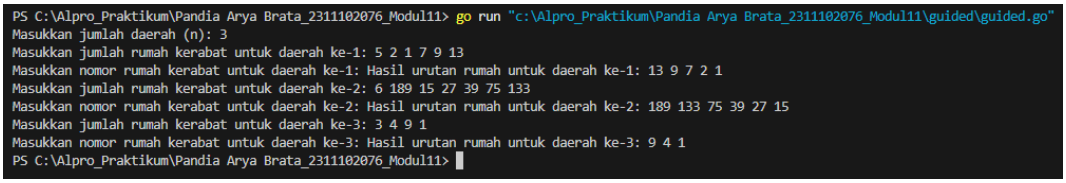
I. GUIDED

1.GUIDED

```
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.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
        }

        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])
        }

        selectionSort(houses)

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

```
            fmt.Printf("%d ", house)
        }
        fmt.Println()
    }
}
```

Screenshot program:



Deskripsi program :

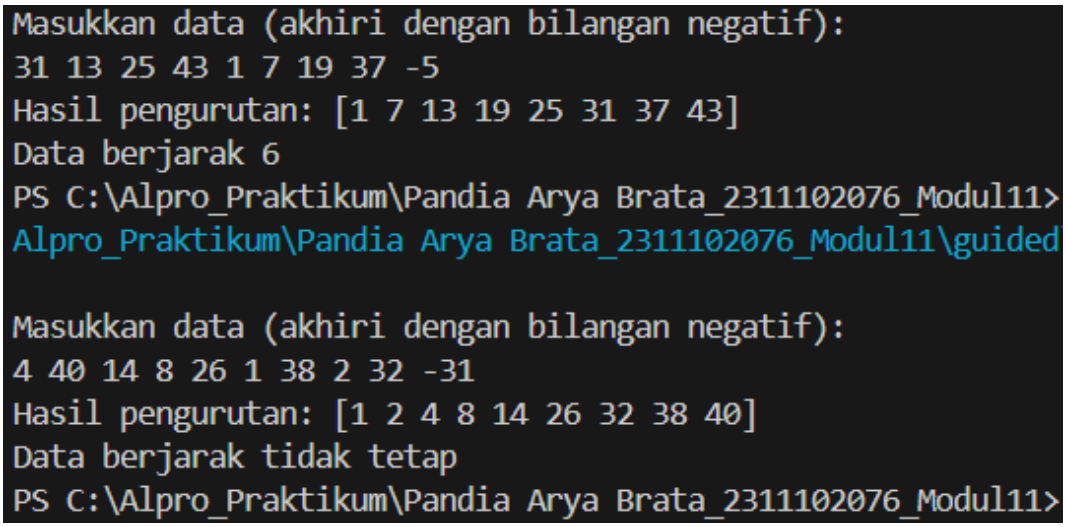
Program tersebut mengurutkan nomor rumah kerabat di beberapa daerah menggunakan algoritma selection short. Pengguna diminta untuk memasukkan jumlah daerah, jumlah rumah di setiap daerah, serta nomor rumah kerabat untuk masing-masing daerah.

2.GUIDED

```
package main

import (
    "fmt"
    "math"
)

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 isDataConsistentlySpaced(arr []int) (bool, int) {
    if len(arr) < 2 {
        return true, 0
    }

    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
        }
    }

    return true, diff
}

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

    fmt.Println("Masukkan data (input negatif untuk
selesai):")
    for {
        fmt.Scan(&input)
        if input < 0 {
            break
        }
        data = append(data, input)
    }

    insertionSort(data)

    isConsistent, diff := isDataConsistentlySpaced(data)

    fmt.Println("Hasil pengurutan:", data)
    if isConsistent {
        fmt.Printf("Data berjarak %d\n", diff)
    } else {
        fmt.Println("Data berjarak tidak tetap")
    }
}
```

Screenshot program :



Deskripsi program :
Program ini menerima input bilangan, mengurutkannya dengan Insertion Sort, dan memeriksa apakah elemen-elemen memiliki jarak tetap. Hasilnya mencakup array terurut dan informasi tentang konsistensi jarak antar elemen.

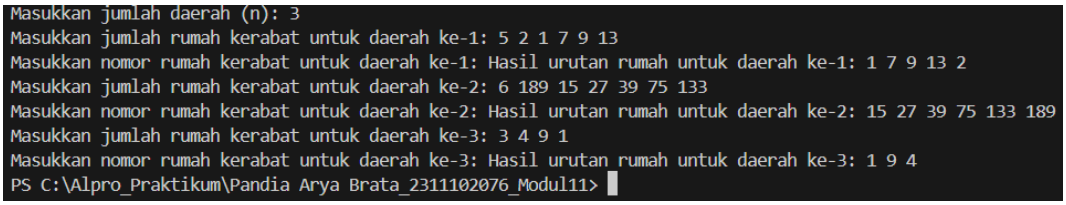
II. UNGUIDED

1.UNGUIDED

```
package main

import (
    "bufio"
    "fmt"
    "os"
    "sort"
    "strconv"
    "strings"
)

func hitungMedian(data []int) int {
    sort.Ints(data)
    n := len(data)
    if n%2 == 0 {
        return (data[n/2-1] + data[n/2]) / 2
    }
    return data[n/2]
}

func main() {
    reader := bufio.NewReader(os.Stdin)
    fmt.Println("Masukkan angka (akhiri dengan -5313):")
    var numbers []int

    for {
        input, _ := reader.ReadString('\n')
        input = strings.TrimSpace(input)

        elements := strings.Split(input, " ")
        for _, elem := range elements {
            num, err := strconv.Atoi(elem)
            if err != nil {
                fmt.Println("Input tidak valid:", elem)
                continue
            }

            if num == -5313 {
                return
```

```
            }

            if num == 0 {
                if len(numbers) > 0 {
                    fmt.Println(hitungMedian(numbers))
                } else {
                    fmt.Println("TIDAK ADA DATA UNTUK
MENGHITUNG MEDIAN")
                }
            } else {
                numbers = append(numbers, num)
            }
        }
    }
}
```

Screenshot program :



Deskripsi program :

Program ini meminta inputan nomor rumah dari beberapa daerah, memisahkannya menjadi dua kategori berdasarkan ganjil atau genap, lalu mengurutkan nomor ganjil dan genap digabungkan dan ditampilkan sebagai hasil urutan untuk setiap daerahnya. Program juga memvalidasi jumlah daerah dan nomor rumah yang dimasukkan agar sesuai dengan batas yang ditentukan.

2.UNGUIDED

```
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 arr []int
    var slice []int
```

```
    var n int

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

    for i := 0; i < panjang; i++ {
        if arr[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, arr[i])
        }
    }

}
```

Screenshot program :

```
7 23 11 0 5 19 2 29 3 13 17 0 -5313
11
12
PS C:\Alpro_Praktikum\Pandia Arya Brata_2311102076_Modul11>
```

Deskripsi program :

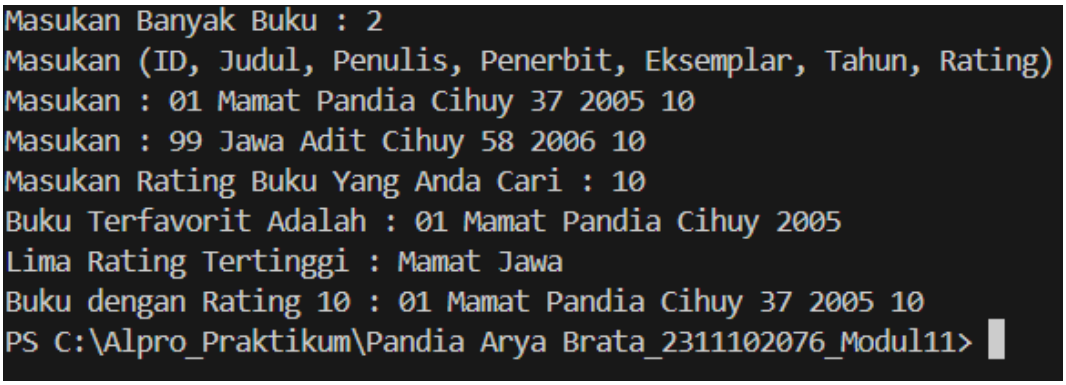
Program ini merupakan implementasi untuk menghitung nilai median dari sekumpulan data bilangan bulat yang diinput secara dinamis oleh pengguna. Setiap kali anga “0” terbaca, program akan menghitung dan mencetak median dari data yang sudah dimasukan sebelumnya, sementara angka “-5313” akan menghentikan program.

3.UNGUIDED

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

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

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

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

    }
    fmt.Println()
}

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

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

        if Pustaka[mid].Ranting == target {
            return mid
        } else if Pustaka[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)

    favorit(Buku, n)
    LimaRating(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 program :



Deskripsi program :

Program ini mengelola data buku dengan memungkinkan pengguna memasukkan informasi buku, kemudian menampilkan buku dengan rating tertinggi, lima buku dengan rating tertinggi, dan mencari buku berdasarkan rating menggunakan algoritma binary search. Pengurutan buku dilakukan dengan insertion sort berdasarkan rating.