LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK

MODUL XII

PENGURUTAN DATA

Oleh:

AULIA RADIX PUTRA WINARKO

2311102056

S1IF-11-02

S1 TEKNIK INRMATIKA

INSTITUT TEKNOLOGI TLKOM PURWOKERTO

2024

I. DASAR TEORI

A. 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	}

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
```

```
    i = 1
    for i <= n-1 {
        idx_min = i - 1
        j = i
        for j < n {
            if T[idx_min] > T[j] {
                idx_min = j
            }
            j = j + 1
        }
        t = T[idx_min]
        T[idx_min] = T[i-1]
        T[i-1] = t
        i = i + 1
    }
}
```

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.

```
...
type mahasiswa struct {
    nama, nim, kelas, jurusan string
    ipk float64
}
type arrMhs [2023]mahasiswa
...
func selectionSort2(T * arrMhs, n int){
/* I.S. terdefinisi array T yang berisi n data mahasiswa
   F.S. array T terurut secara asceding atau membesar berdasarkan ipk dengan
menggunakan algoritma SELECTION SORT */
    var i, j, idx_min int
    var t mahasiswa
    i = 1
    for i <= n-1 {
        idx_min = i - 1
        j = i
        for j < n {
            if T[idx_min].ipk > T[j].ipk {
                idx_min = j
            }
            j = j + 1
        }
        t = T[idx_min]
        T[idx_min] = T[i-1]
        T[i-1] = t
        i = i + 1
    }
}
```

II. 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()
    }
}
```

Output

```
PS C:\Users\user\go\src\Praktikum Alpro2\modul 12> go run "c:\U
c:\Users\user\go\src\Praktikum Alpro2\modul 12\GUIDED\main.go"
Masukkan jumlah daerah (n): 3
Masukkan jumlah rumah kerabat untuk daerah ke-1: 5
Masukkan nomor rumah kerabat untuk daerah ke-1: 34
23
56
34
54
Hasil urutan rumah untuk daerah ke-1: 56 54 34 34 23
Masukkan jumlah rumah kerabat untuk daerah ke-2: 2
Masukkan nomor rumah kerabat untuk daerah ke-2: 23
44
Hasil urutan rumah untuk daerah ke-2: 44 23
Masukkan jumlah rumah kerabat untuk daerah ke-3: 44
Masukkan nomor rumah kerabat untuk daerah ke-3: 23
```

Keterangan

Program ini menggunakan algoritma pengurutan pilihan untuk mengurutkan nomor rumah relatif di beberapa area dari yang tertinggi hingga terendah. User menginput jumlah luas (n) dan

jumlah rumah pada setiap luas (m), kemudian memasukkan jumlah rumah pada setiap luas. Setelah diurutkan, program akan mencetak alamat setiap area yang diurutkan. Jika input n atau m tidak valid, program menampilkan kesalahan dan berhenti.

2. 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()
    }
}
```

Output

```
PS C:\Users\user\go\src\Praktikum Alpro2\modul 12> go ru
Masukkan data (akhiri dengan bilangan negatif):
33 13 25 43 1 7 19 37 -5
Hasil pengurutan: [1 7 13 19 25 33 37 43]
Data berjarak tidak tetap
PS C:\Users\user\go\src\Praktikum Alpro2\modul 12>
```

Keterangan

Program ini menerima sekumpulan angka dari user, mengurutkannya menggunakan algoritma pengurutan penyisipan, dan memeriksa apakah jarak antara elemen array yang diurutkan adalah konstan. Jika jaraknya tetap, program akan menampilkan jaraknya. Jika tidak, program menunjukkan bahwa jaraknya tidak tetap. Jika pengguna memasukkan angka negatif, input dihentikan.

III. UNGUIDED

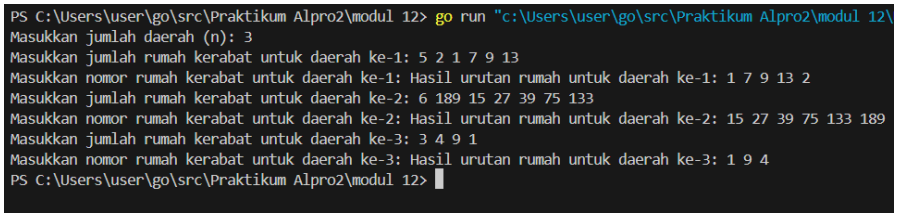
1. Source Code

```
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 selectionSortDesc(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])
        }

        var oddNumbers, evenNumbers []int
        for _, house := range houses {
            if house%2 == 1 {
                oddNumbers =
append(oddNumbers, house)
            } else {
                evenNumbers =
append(evenNumbers, house)
            }
        }

        selectionSortAsc(oddNumbers)

        selectionSortDesc(evenNumbers)

        fmt.Printf("Hasil urutan rumah
untuk daerah ke-%d: ", i+1)

        for _, house := range oddNumbers {
            fmt.Printf("%d ", house)
        }

        for _, house := range evenNumbers {
            fmt.Printf("%d ", house)
        }
        fmt.Println()
```

```
    }
}
```

Output



Keterangan

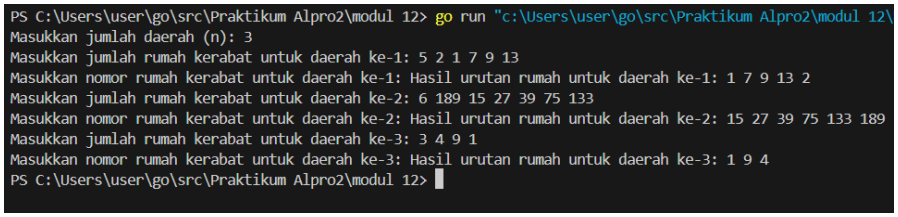
Program di atas adalah implementasi dalam bahasa Go untuk mengurutkan nomor rumah berdasarkan beberapa kriteria. Pengguna diminta memasukkan jumlah daerah (n) dan jumlah rumah kerabat (m) untuk setiap daerah, kemudian memasukkan daftar nomor rumah. Nomor rumah dikelompokkan menjadi bilangan ganjil dan genap, lalu masing-masing diurutkan: ganjil secara ascending (kecil ke besar) dan genap secara descending (besar ke kecil)

2. Source Code

```
package main

import "fmt"

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

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 = j - 1
        }
        arr[j+1] = key
    }
}
```

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

    for {
        fmt.Scan(&input)
        if input == -5313 {
            break
        }

        if input == 0 {
            insertionSort(data)
            median := hitungMedian(data)
            fmt.Println(median)
        } else {
            data = append(data, input)
        }
    }
}
```

Output



Keterangan

Program di atas adalah implementasi dalam bahasa Go untuk menghitung median dari sekumpulan data yang dimasukkan secara bertahap oleh pengguna. Program membaca bilangan secara terus-menerus sampai pengguna memasukkan nilai -5313, yang digunakan sebagai penanda untuk mengakhiri input.

3. Source Code

```
package main

import (
    "fmt"
)

const nMax = 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("Buku %v: \n",i+1)
    fmt.Print("Masukkan id buku: ")
    fmt.Scan(&pustaka[i].id)
    fmt.Print("Masukkan judul buku: ")
    fmt.Scan(&pustaka[i].judul)
    fmt.Print("Masukkan penulis buku: ")
    fmt.Scan(&pustaka[i].penulis)
    fmt.Print("Masukkan penerbit buku: ")
    fmt.Scan(&pustaka[i].penerbit)
    fmt.Print("Masukkan eksemplar buku: ")
    fmt.Scan(&pustaka[i].eksemplar)
    fmt.Print("Masukkan tahun terbit: ")
    fmt.Scan(&pustaka[i].tahun)
    fmt.Print("Masukkan rating buku: ")
    fmt.Scan(&pustaka[i].rating)
    fmt.Println()
  }
}

func CetakTerfavorit(pustaka DaftarBuku ,n
int){
  max := pustaka[0]
  for i := 1; i < n; i++ {
    if pustaka[i].rating > max.rating{
      max = pustaka[i]
    }
  }
  fmt.Println("Buku Favorit: ")
  fmt.Printf("Judul buku: %v\n", max.judul)
  fmt.Printf("Penulis buku: %v\n",
max.penulis)
  fmt.Printf("Penerbit buku: %v\n",
max.penerbit)
  fmt.Println()
}

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){
  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
  }
  fmt.Println("5 Judul Buku dengan Rating
Tertinggi: ")
  limit := 5
  if n < limit {
    limit = n
  }
  for i := 0; i < limit; i++ {
    fmt.Printf("Buku %v: \n",i+1)
    fmt.Printf("Judul buku: %v\n",
pustaka[i].judul)
    fmt.Println()
  }
}

func CariBuku(pustaka DaftarBuku, n, r int)
{
    fmt.Print("Masukkan rating untuk
mencari buku: ")
    fmt.Scan(&r)

    low, high := 0, n-1
```

```
    found := false
    for low <= high {
        mid := (low + high) / 2
        if pustaka[mid].rating == r {
            fmt.Println("Buku dengan
rating", r, ":")

            i := mid
            for i >= 0 && pustaka[i].rating
== r {
                fmt.Printf("Buku %v: \n",
i+1)
                fmt.Printf("Judul buku:
%v\n", pustaka[i].judul)
                fmt.Printf("Penulis buku:
%v\n", pustaka[i].penulis)
                fmt.Printf("Penerbit buku:
%v\n", pustaka[i].penerbit)
                fmt.Printf("Tahun terbit:
%v\n", pustaka[i].tahun)
                fmt.Println()
                i--
            }

            i = mid + 1
            for i < n && pustaka[i].rating
== r {
                fmt.Printf("Buku %v: \n",
i+1)
                fmt.Printf("Judul buku:
%v\n", pustaka[i].judul)
                fmt.Printf("Penulis buku:
%v\n", pustaka[i].penulis)
                fmt.Printf("Penerbit buku:
%v\n", pustaka[i].penerbit)
                fmt.Printf("Tahun terbit:
%v\n", pustaka[i].tahun)
                fmt.Println()
                i++
            }
            found = true
            break
        } else if pustaka[mid].rating < r {
            low = mid + 1
        } else {
            high = mid - 1
        }
```

```
    }
    if !found {
        fmt.Println("Tidak ada buku dengan
rating seperti itu")
    }
}

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

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

  DaftarkanBuku(&pustaka, Npustaka)
  CetakTerfavorit(pustaka, Npustaka)
  UrutBuku(pustaka,Npustaka)
  Cetak5Terbaru(pustaka,Npustaka)
  CariBuku(pustaka,Npustaka,rating)
}
```

Output

```
PS C:\Users\user\go\src\Praktikum Alpro2\modul 12> go run "
c:\Users\user\go\src\Praktikum Alpro2\modul 12\UNGUIDED\tem
pCodeRunnerFile.go"
Masukkan jumlah buku: 1
Buku 1:
Masukkan id buku: 1
Masukkan judul buku: Si_kancil
Masukkan penulis buku: Andi
Masukkan penerbit buku: Pustaka_Buku
Masukkan eksemplar buku: 6
Masukkan tahun terbit: 2023
Masukkan rating buku: 6

Buku Favorit:
Judul buku: Si_kancil
Penulis buku: Andi
Penerbit buku: Pustaka_Buku

5 Judul Buku dengan Rating Tertinggi:
Buku 1:
Judul buku: Si_kancil

Masukkan rating untuk mencari buku:
```

Keterangan

Program di atas adalah implementasi sistem manajemen buku sederhana menggunakan bahasa Go. Program ini menangani berbagai operasi terkait daftar buku, termasuk pendaftaran, pencarian, pengurutan, dan pencetakan buku berdasarkan kriteria tertentu.