LAPORAN PRAKTIKUM
ALGORITMA PEMROGRAMAN 2

MODUL 12 & 13

PENGURUTAN DATA

Oleh:

WILDAN DAFFA' HAKIM PUTRA ANTARA

2311102055

S1IF-11-02

S1 TEKNIK INFORMATIKA

UNIVERSITAS TELKOM

PURWOKERTO

2024

I. DASAR TEORI

1. Selection Sort

   Pada selection sort, pengurutan dilakukan dengan mencari nilai ekstrim (minimal/maksimal). Lalu ditaruh pada indeks yang lebih kiri. Langkah – langkahnya sebagai berikut :

   1. Cari nilai ekstrim dari data
   2. Pindahkan/tukar tempat dengan data yang berada pada posisi paling kiri.
   3. Ualngi perulangan sampai hanya tersisa satu data.

Dalam Pseudocode	Dalam Golang

`i ← 1`
`while i <= n-1 do`
`  idx_min ← i - 1`
`  j ← i`
`  while j < n do`
`  if a[idx_min] > a[j] then`
`  idx_min ← j`
`  endif`
`  j ← j + 1`
`  endwhile`
`  t ← a[idx_min]`
`  a[idx_min] ← a[i-1]`
`  a[i-1] ← t`
`  i ← i + 1`
`endwhile`
	`i = 1`
	`for i <= n-1 {`
	`  idx_min = i - 1`
	`  j = i`
	`  for j < n {`
	`  if a[idx_min] > a[j] {`
	`  idx_min = j`
	`  }`
	`  j = j + 1`
	`  }`
	`  t = a[idx_min]`
	`  a[idx_min] = a[i-1]`
	`  a[i-1] = t`
	`  i = i + 1`
	`}`

2. Insertion Sort

   Metode pengurutan ini memiliki cara yang berbeda dengan selection sort dimana menempatkan angaka pada posisi yang sesuai menggunakan sequential search. Langkah – langkahnya :

   1. Ambil angka yang manu dicari posisinya pada data terurut.
   2. Geser data ke kanan sehingga menyisakan satu tempat kosong untuk memasukkan data yang belum terurut ke posisinya tanpa merubah data yang sudah terurut
   3. Ulangi sampai samua terurut

Dalam Pseudocode	Dalam Golang
`i ← 1` `while i <= n-1 do`	`i = 1` `for i <= n-1 {`

```
    j ← i
    temp ← a[j]
    while j > 0 and temp >
a[j-1] do
        a[j] ← a[j-1]
        j ← j - 1
    endwhile
    a[j] ← temp
    i ← i + 1
endwhile
```

```
    j = i
    temp = a[j]
    for j > 0 && temp >
a[j-1] {
        a[j] = a[j-1]
        j = j - 1
    }
    a[j] = temp
    i = i + 1
}
```

II. GUIDED

1. Hercules, preman terkenal seantero ibukota, memiliki kerabat di banyak daerah. Tentunya Hercules sangat suka mengunjungi semua kerabatnya itu..

```
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] {
                        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
            }

            // 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()
      }
}
```

```
12
λ MacBook-Air-Daffa Modul 12 n 13 → go run "/Users/daffahakim/Documents/Kuliah/SMT 3/Alpro
Masukkan jumlah daerah (n): 3
Masukkan jumlah rumah kerabat untuk daerah ke-1: 5
Masukkan nomor rumah kerabat untuk daerah ke-1: 2 3 5 9 1
Hasil urutan rumah untuk daerah ke-1: 9 5 3 2 1
Masukkan jumlah rumah kerabat untuk daerah ke-2: 6
Masukkan nomor rumah kerabat untuk daerah ke-2: 3 2 6 9 1 4
Hasil urutan rumah untuk daerah ke-2: 9 6 4 3 2 1
Masukkan jumlah rumah kerabat untuk daerah ke-3: 4
Masukkan nomor rumah kerabat untuk daerah ke-3: 2 4 3 1
Hasil urutan rumah untuk daerah ke-3: 4 3 2 1
```

Program ini digunakan untuk mengurutkan nomor rumah kerabat Hercules pada setiap daerah secara descending. program dimulai dengan meminta pengguna untuk memasukkan jumlah daerah yang ingin diolah. Program memvalidasi agar jumlah daerah tersebut lebih besar dari 0 dan kurang dari 1000. Selanjutnya, untuk setiap daerah, pengguna diminta untuk menginput jumlah rumah kerabat, dengan validasi bahwa nilainya harus lebih besar dari 0 dan kurang dari 1.000.000.

Setelah itu, pengguna diminta memasukkan nomor rumah kerabat dalam bentuk array. Program kemudian menggunakan fungsi selectionSort untuk mengurutkan nomor rumah tersebut secara menurun. Fungsi ini bekerja dengan mencari elemen terbesar dalam array pada setiap iterasi dan menukarnya ke posisi awal. Setelah proses

pengurutan selesai, hasilnya dicetak ke output yang menghasilkan nomor rumah kerabat dengan urutan descending.

2. Buatlah sebuah program yang digunakan untuk membaca data integer seperti contoh yang diberikan di bawah ini, kemudian diurutkan (menggunakan metoda insertion sort), dan memeriksa apakah data yang terurut berjarak sama terhadap data sebelumnya.

```
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")
	}
}
```

```
λ MacBook-Air-Daffa Modul 12 n 13 → go run "/Users/daffahakim/Documents/Kulia
Masukkan data (akhiri dengan bilangan negatif):
31 13 25 43 1 7 19 37 -5
Hasil pengurutan: [1 7 13 19 25 31 37 43]
Data berjarak 6
λ MacBook-Air-Daffa Modul 12 n 13 → go run "/Users/daffahakim/Documents/Kulia
Masukkan data (akhiri dengan bilangan negatif):
4 40 14 8 26 1 38 2 32 -31
Hasil pengurutan: [1 2 4 8 14 26 32 38 40]
Data berjarak tidak tetap
λ MacBook-Air-Daffa Modul 12 n 13 →
```

Program ini digunakan untuk mengurutkan angka secara menaik (*ascending*) dan memeriksa apakah bilangan-bilangan tersebut memiliki jarak tetap. Pada subprogram main, dideklarasikan variabel data berupa slice dengan tipe data integer untuk menyimpan bilangan yang diinput pengguna, serta variabel input untuk menampung nilai input sementara. Program meminta pengguna untuk memasukkan bilangan satu per satu hingga pengguna memasukkan bilangan negatif, yang menjadi tanda akhir input. Semua bilangan yang dimasukkan ditambahkan ke dalam slice data dengan for loop.

Setelah seluruh input diterima, slice data diurutkan menggunakan fungsi insertionSort. Fungsi ini bekerja dengan membandingkan setiap elemen baru dengan elemen-elemen sebelumnya dalam slice, lalu menempatkannya pada posisi yang sesuai agar array terurut secara menaik.

Setelah pengurutan selesai, program memanfaatkan fungsi isDataConsistentlySpaced untuk memeriksa apakah elemen-elemen dalam array memiliki jarak tetap. Fungsi ini menghitung selisih absolut antara dua elemen pertama, kemudian membandingkannya dengan selisih antar elemen lainnya. Jika semua selisih sama, data dianggap memiliki jarak tetap, dan program mencetak nilai selisih tersebut. Jika tidak, program memberikan output bahwa data tidak memiliki jarak tetap.

III. UNGUIDED

1. Belakangan diketahui ternyata Hercules itu tidak berani menyeberang jalan, maka selalu diusahakan agar hanya menyeberang jalan sesedikit mungkin, hanya diujung jalan. Karena nomor rumah sisi kiri jalan selalu ganjil dan sisi kanan jalan selalu genap, maka buatlah program kerabat dekat yang akan menampilkan nomor rumah mulai dari nomor yang ganjil lebih dulu terurut membesar dan kemudian menampilkan nomor rumah dengan nomor genap terurut mengecil.

```
package main

import "fmt"

func selectionSortGanjil(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] //
Tukar elemen
    }
}

func selectionSortGenap(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] // Tukar elemen
    }
}

func main() {
    var n, input 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
        housesGanjil := make([]int, 0)
        housesGenap := make([]int, 0)
        fmt.Printf("Masukkan nomor rumah kerabat untuk daerah ke-%d: ", i+1)
        for j := 0; j < m; j++ {
            fmt.Scan(&input)
            if input%2 == 0 {
                housesGenap    =    append(housesGenap, input)
            } else {
                housesGanjil   =   append(housesGanjil, input)
            }
        }

        // Urutkan dengan selection sort
        selectionSortGanjil(housesGanjil)
        selectionSortGenap(housesGenap)

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

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

```
λ MacBook-Air-Daffa Modul 12 n 13 → go run "/Users/daffahakim/Documents/Kuliah/SM
Masukkan jumlah daerah (n): 3
Masukkan jumlah rumah kerabat untuk daerah ke-1: 5
Masukkan nomor rumah kerabat untuk daerah ke-1: 2 4 1 3 5
Hasil urutan rumah untuk daerah ke-1: 5 3 1 2 4
Masukkan jumlah rumah kerabat untuk daerah ke-2: 6
Masukkan nomor rumah kerabat untuk daerah ke-2: 7 9 3 2 4 6
Hasil urutan rumah untuk daerah ke-2: 9 7 3 2 4 6
Masukkan jumlah rumah kerabat untuk daerah ke-3: 5
Masukkan nomor rumah kerabat untuk daerah ke-3: 1 2 5 7 9
Hasil urutan rumah untuk daerah ke-3: 9 7 5 1 2
```

Program ini merupakan perubahan dari program sebelumnya. Secara inputan masih sama seperti program sebelumnya. Hanya saja Ketika angka diinputkan akann dilakukan pengecekan apakah bilangan tersebut genap atau ganjil. Jika genap maka akan masuk slice dengan nama housesGenap dan jika ganjil maka akan masuk slice housesGanjil. Selesai mengelompokkan input dilakukan selection sort kepada kedua slice ini, untuk slice yang berisi bilangan ganjil diurutkan secara descending, sedangakan untuk yang genap diurutkan secara ascending. Setelah diurutkan data dicetak dengan urutan ganjil terlebih dahulu baru genap

2. Kompetisi pemrograman yang baru saja berlalu diikuti oleh 17 tim dari berbagai perguruan tinggi ternama. Dalam kompetisi tersebut, setiap tim berlomba untuk menyelesaikan sebanyak mungkin problem yang diberikan. Dari 13 problem yang diberikan, ada satu problem yang menarik. Problem tersebut mudah dipahami, hampir semua tim mencoba untuk menyelesaikannya, tetapi hanya 3 tim yang berhasil. Apa sih problemnya?

   Buatlah program median yang mencetak nilai median terhadap seluruh data yang sudah terbaca, jika data yang dibaca saat itu adalah 0.

```
package main

import "fmt"

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
      }
}

func median(arr []int) int {
      n := len(arr)
      if n%2 == 0 {
            return (arr[(n/2)-1] + arr[(n/2)]) / 2

      } else {

            return arr[(n / 2)]
      }
}

func main() {
      var input int
      var kumpulanAngka = make([]int, 0)
      var angkaSliced = make([]int, 0)
      for input != -5313 {
            fmt.Scan(&input)
            if input != -5313 {
                  kumpulanAngka = append(kumpulanAngka,
input)
            }
      }
      for _, angka := range kumpulanAngka {
            if angka == 0 {
                  insertionSort(angkaSliced)
                  fmt.Println(median(angkaSliced))

            } else {
                  angkaSliced   =   append(angkaSliced,
angka)
            }
      }
}
```

```
λ MacBook-Air-Daffa Modul 12 n 13 → go run "/Users/daffaha
7 23 11 0 5 19 2 29 3 13 17 0 -5313
11
12
λ MacBook-Air-Daffa Modul 12 n 13 →
```

Program ini digunakan untuk mecari median setiap muncul angka nol. Pada subprogram main dideklarasikan variable input bertipe integer,

kumpulanAngka dan angkaSliced yang merupakan array slice dengan tipe integer. Setelah itu, dilakukan perulangan dimana pengguna diminta menginputkan angka selain -5313 dan akan disimpan pada slice kumpulanAngka.

Setelah perulangan selesai, dilakukan lagi perulangan dengan for – range, jika nilai angka tidak sama dengan nol maka nilai angka dimasukkan pada slice angkaSliced. Jika angka sama dengan nol maka slice angkaSliced akan diurutkan setelah itu mediannya akan dicetak.

3. Sebuah program perpustakaan digunakan untuk mengelola data buku di dalam suatu perpustakaan. Misalnya terdefinisi struct dan array seperti berikut ini:

```
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) {
      fmt.Println("Format masukan (id, judul, penulis,
penerbit, eksamplar, tahun, rating)")
      for i := 0; i < n; i++ {
            fmt.Print("Buku ke ", i+1, " : ")
            fmt.Scanln(&pustaka[i].id,
&pustaka[i].judul,                  &pustaka[i].penulis,
&pustaka[i].penerbit,             &pustaka[i].eksemplar,
&pustaka[i].tahun, &pustaka[i].rating)
      }
}

func CetakTerfavorit(pustaka DaftarBuku, n int) {
      var indexFav = 0
      for i := 1; i < n; i++ {
            if            pustaka[i].rating           >
pustaka[indexFav].rating {
                  indexFav = i
            }
      }

      fmt.Printf("Buku terfavorit sekarang: : %s %s %s
%s %d %d %d\n",
```

```
            pustaka[indexFav].id,
pustaka[indexFav].judul,    pustaka[indexFav].penerbit,
pustaka[indexFav].penulis,
            pustaka[indexFav].eksemplar,
pustaka[indexFav].rating, pustaka[indexFav].tahun)
}

func UrutBuku(pustaka *DaftarBuku, n int) {
      for i := 1; i < n; i++ {
            var key Buku = pustaka[i]
            var j int = 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) {
      if n > 5 {
            n = 5
      }
      fmt.Print(n, " buku rating tertinggi :")
      for i := 0; i < n; i++ {
            fmt.Print(" ", pustaka[i].judul)
      }
      fmt.Println()
}

func CariBuku(pustaka DaftarBuku, n, r int) {
      kr := 0
      kn := n - 1
      var med int
      var found bool = false

      for kr <= kn && !found {
            med = (kr + kn) / 2

            if pustaka[med].rating > r {
                  kr = med + 1
            } else if pustaka[med].rating < r {
                  kn = med - 1
            } else {
                  found = true
            }
      }
      if found {
            fmt.Printf("Buku dengan rating %d: %s %s %s
%s %d %d %d\n",
                  r,                    pustaka[med].id,
pustaka[med].judul,               pustaka[med].penerbit,
pustaka[med].penulis,
                  pustaka[med].eksemplar,
pustaka[med].rating, pustaka[med].tahun)
```

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

func main() {
	var pustaka DaftarBuku
	var nPustaka, chooseRating int
	fmt.Print("Masukkan banyaknya buku : ")
	fmt.Scan(&nPustaka)
	DaftarkanBuku(&pustaka, nPustaka)
	fmt.Print("Masukkan rating buku yang dicari : ")
	fmt.Scan(&chooseRating)
	CetakTerfavorit(pustaka, nPustaka)
	UrutBuku(&pustaka, nPustaka)
	Cetak5Terbaru(pustaka, nPustaka)
	CariBuku(pustaka, nPustaka, chooseRating)
}
```

```
λ MacBook-Air-Daffa Modul 12 n 13 → go run "/Users/daffahakim/Documents/Kuliah/SMT 3/Alpro2/
Masukkan banyaknya buku : 7
Format masukan (id, judul, penulis, penerbit, eksamplar, tahun, rating)
Buku ke 1 : B001 Cerah Ali Gramedia 10 2021 4
Buku ke 2 : B002 Hujan Budi Erlangga 15 2020 3
Buku ke 3 : B003 Bumi Citra Balai 8 2019 4
Buku ke 4 : B004 Langit Dian Mizan 12 2022 2
Buku ke 5 : B005 Laut Eka Bentang 7 2018 3
Buku ke 6 : B006 Angin Fina Grasindo 20 2023 5
Buku ke 7 : B007 Bintang Hani Kanisius 13 2020 4
Masukkan rating buku yang dicari : 2
Buku terfavorit sekarang: : B006 Angin Grasindo Fina 20 5 2023
5 buku rating tertinggi : Angin Cerah Bumi Bintang Hujan
Buku dengan rating 2: B004 Langit Mizan Dian 12 2 2022
λ MacBook-Air-Daffa Modul 12 n 13 →
```

Program ini dibuat untuk melakukan pencarian buku favorit, menampilkan 5 buku dengan rating tertinggi, serta menampilkan buku berdasarkan rating yang ditentukan oleh pengguna. Di dalam subprogram main, variabel pustaka dideklarasikan sebagai array dengan kapasitas maksimal sebesar nilai nMax (7919), sedangkan jumlah buku yang akan diinputkan ditentukan melalui masukan pengguna dan disimpan dalam variabel nPustaka.

Pengguna diminta memasukkan data buku satu per satu sesuai format yang telah ditentukan sebanyak nPustaka, dan data tersebut disimpan dalam array pustaka. Selanjutnya, pengguna diminta memasukkan rating yang ingin dicari. Program kemudian memanggil prosedur CetakTerfavorit, yang mencari buku dengan rating tertinggi menggunakan metode sequential search. Setelah itu, dipanggil prosedur UrutBuku yang akan mengurutkan buku-buku dalam array berdasarkan rating secara descending menggunakan insertion sort. Setelah diurutkan, dipanggil prosedur Cetak5Terbaru yang akan menampilkan hingga 5

buku pertama (jika jumlahnya >= 5), yang merupakan buku dengan rating tertinggi.

Setelah itu, dipangil prosedur CariBuku yang akan mencari buku dengan rating tertentu menggunakan binary search pada array yang telah diurutkan. Jika buku ditemukan, program mencetak detail buku tersebut; jika tidak, program memberikan pesan bahwa tidak ada buku dengan rating yang sesuai.