LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL XII

PENGURUTAN DATA

Disusun Oleh :

Avrizal Setyo Aji Nugroho

2311102145

IF-11-05

Dosen Pengampu :

Arif Amrulloh, S.Kom., M.Kom

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Pengurutan data adalah proses mengatur elemen dalam kumpulan data berdasarkan urutan tertentu, seperti menaik (ascending) atau menurun (descending), berdasarkan kriteria tertentu, seperti angka, abjad, atau atribut lainnya. Tujuan dari pengurutan data adalah untuk membuat pencarian, analisis, dan pengelolaan data lebih mudah, terutama untuk kumpulan data yang sangat besar.
Selection Sort adalah Algoritma pengurutan sederhana memilih elemen terkecil dari bagian array yang belum diurutkan dan menukarnya dengan elemen di posisi awal bagian tersebut. Proses ini diulangi untuk setiap elemen, sehingga bagian array yang terurut bertambah satu elemen setiap langkahnya. Sampai seluruh elemen array terurut, algoritma ini terus mencari nilai terkecil.

Insertion Sort adalah algoritma pengurutan yang mengambil satu elemen dari bagian array yang belum diurutkan dan menyisipkannya ke dalam posisi yang tepat di bagian array yang sudah diurutkan. Prosedur ini dilakukan berulang kali, dimulai dari elemen pertama, hingga semua elemen berada di posisi yang benar. Setiap kali sebuah elemen baru dipertimbangkan, elemen di bagian yang sudah terurut dapat digeser untuk memberikan tempat bagi elemen yang sedang dimasukkan. Karena lebih cepat dalam situasi seperti itu, insertion0020sort biasanya digunakan untuk dataset kecil atau data yang sudah hampir terurut.

II. GUIDED

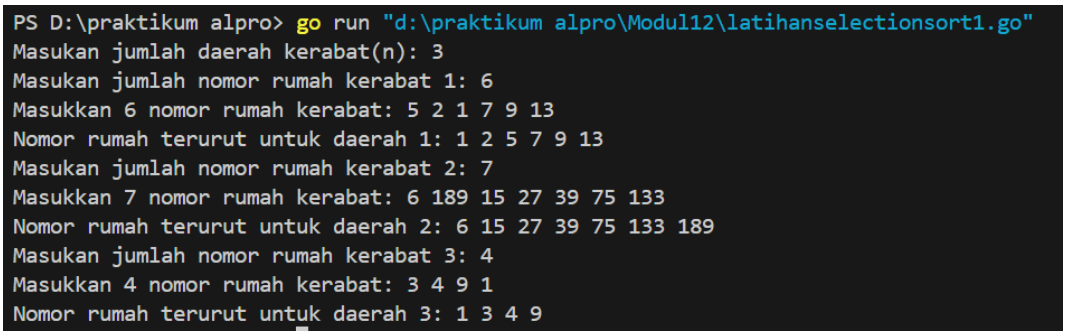
1. Selection Sort

   Sourcecode

```
package main

import (
   "fmt"
)

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

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

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

         arr := make([]int, m)
         fmt.Printf("Masukkan %d nomor rumah kerabat:
", m)
         for i := 0; i < m; i++ {
```

```
                fmt.Scan(&arr[i])
            }
            selectionSort(arr, m)

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

Screenshoot Output



Deskripsi Program

Program di atas menggunakan algoritma selection sort untuk mengurutkan daftar nomor rumah kerabat dalam beberapa daerah. Program meminta pengguna untuk memasukkan jumlah daerah kerabat (n), kemudian memasukkan jumlah rumah (m) dan nomor rumah untuk setiap daerah. Algoritma sort pilihan akan digunakan untuk mengurutkan nomor rumah secara menaik. Algoritma ini bekerja dengan memilih elemen terkecil dalam array dan menukarnya dengan elemen di posisi tersebut. Program akan mencetak daftar nomor rumah untuk setiap daerah setelah semua nomor rumah diurutkan.

2. Insertion Sort

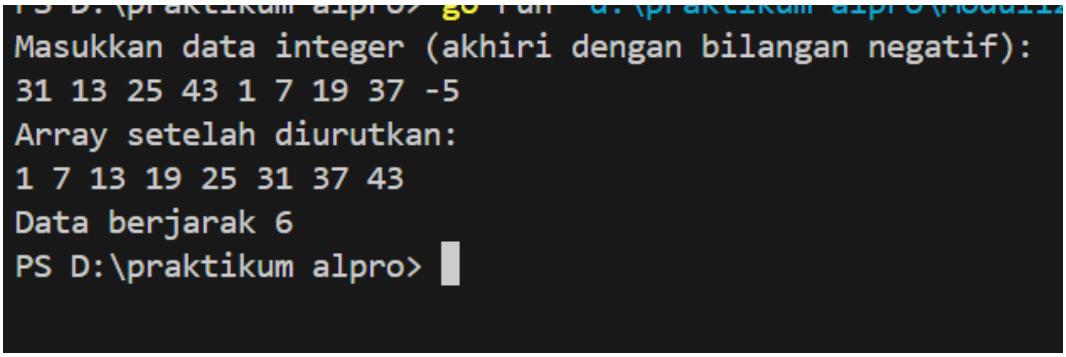
Sourcecode

```
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")
      }
}
```

Screenshoot Output



Deskripsi Program

Program di atas menggunakan algoritma insertion sort untuk mengurutkan sebuah array angka bulat (integer) yang dimasukkan oleh pengguna hingga bilangan negatif ditemukan. Setelah pengurutan, program melihat apakah array memiliki selisih tetap antara elemen yang berurutan. Hal ini dicapai dengan algoritma insertion sort, yang memproses elemen array satu per satu, dan menyisipkan setiap elemen yang belum diurutkan pada posisi yang tepat di bagian array yang sudah diurutkan. Selanjutnya, fungsi **isConstantDifference** menentukan apakah selisih antara masing-masing elemen berturut-turut sama atau tidak. Jika jawabannya ya, program akan menampilkan nilai perbedaan.

III. UNGUIDED

1. **Sourcecode selection sort**

```
      package main
```

```
import "fmt"

func selectionSortAscending(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 selectionSortDescending(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("Masukan jumlah daerah kerabat (n):
")
	fmt.Scanln(&n)

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

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

		var ganjil, genap []int
		for _, num := range arr {
			if num%2 == 0 {
```

```
                        genap = append(genap, num)
                    } else {
                        ganjil = append(ganjil, num)
                    }
                }

                selectionSortAscending(ganjil)
                selectionSortDescending(genap)

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

Screenshoot Output

```
PS D:\praktikum alpro> go run "d:\praktikum alpro\Modul12\Unguided1.go"
Masukan jumlah daerah kerabat (n): 3
Masukan jumlah nomor rumah kerabat 1: 6
Masukkan 6 nomor rumah kerabat: 5 2 1 7 9 13
Nomor rumah terurut untuk daerah 1: 1 5 7 9 13 2
Masukan jumlah nomor rumah kerabat 2: 7
Masukkan 7 nomor rumah kerabat: 6 189 15 27 39 75 133
Nomor rumah terurut untuk daerah 2: 15 27 39 75 133 189 6
Masukan jumlah nomor rumah kerabat 3: 4
Masukkan 4 nomor rumah kerabat: 3 4 9 1
Nomor rumah terurut untuk daerah 3: 1 3 9 4
```

Deskripsi Program

Program di atas dirancang untuk mengurutkan nomor rumah kerabat di berbagai area, dengan nomor rumah dengan angka ganjil diurutkan secara menaik (ascending) dan nomor rumah dengan angka genap diurutkan secara menurun (descending). Algoritma Sorting Selection mencari elemen terkecil (ascending) atau terbesar (descending) di setiap iterasi dan kemudian menukarnya dengan elemen yang ada di posisi iterasi saat ini. Program dimulai dengan meminta berapa banyak daerah dan nomor rumah

yang ada di dalamnya. Kemudian, nomor rumah tersebut dibagi menjadi dua kelompok, ganjil dan genap. Kelompok ganjil diurutkan secara menaik menggunakan fungsi selectionSortAscending dan selectionSortDescending.

2. **Sourcecode selection sort**

```
package main

import "fmt"

func selectionsort(arr []int) {
   for i := 0; i < len(arr)-1; i++ {
         minIndex := i
         for j := i + 1; j < len(arr); j++ {
               if arr[j] < arr[minIndex] {
                     minIndex = j
               }
         }
         arr[i], arr[minIndex] = arr[minIndex], arr[i]
   }
}

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

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

   fmt.Println("Input data (akhiri dengan -5313):")

   for {
         fmt.Scan(&n)
         if n == -5313 {
               break
         }
         if n == 0 {
               if len(data) > 0 {
                     selectionsort(data)
                     fmt.Println(median(data))
                     data = []int{}
               }
         } else {
               data = append(data, n)
         }
   }
}
```

Screenshoot Output

```
PS D:\praktikum alpro> go run "d:\praktikum alpro\Modul12\Unguided2.go"
Masukkan data (akhiri dengan -5313):
7 23 11 0 5 19 2 29 3 13 17 0 -5313
11
13
```

Deskripsi Program

Program di atas dirancang untuk menghitung "median" dari sekumpulan bilangan bulat positif yang dimasukkan secara bertahap oleh pengguna. Untuk mengurutkan data, program ini menggunakan algoritma "Selection Sort". Nilai median dari kumpulan data yang sudah terurut dikenal sebagai median. Untuk kumpulan data ganjil, median adalah elemen di tengah, dan untuk kumpulan data genap, median adalah rata-rata dari dua elemen tengah yang dibulatkan ke bawah. Untuk menghitung median dari data yang sudah dimasukkan, pengguna memasukkan bilangan satu per satu, dengan angka "0" sebagai pemicu. Kemudian, data direset. Ketika angka "-5313" dimasukkan, program berakhir. Ini mencetak hasil median setiap kali angka "0" dimasukkan, dan seluruh data sebelumnya telah diurutkan menggunakan "Selection Sort".

3. **Sourcecode insertion sort**

```
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("Masukkan jumlah buku:")
    fmt.Scan(n)
```

```
    for i := 0; i < *n; i++ {
        fmt.Printf("\nMasukkan 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 dalam pustaka.")
        return
    }
    maxRating := pustaka[0].rating
    maxIndex := 0
    for i := 1; i < n; i++ {
        if pustaka[i].rating > maxRating {
            maxRating = pustaka[i].rating
            maxIndex = i
        }
    }
    buku := pustaka[maxIndex]
    fmt.Printf("Buku terfavorit:\nJudul: %s, Penulis:
%s, Penerbit: %s, Tahun: %d\n", buku.judul,
buku.penulis, buku.penerbit, buku.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 := 0
    for i := 0; i < n && count < 5; i++ {
```

```
            fmt.Printf("%d. %s\n", count+1,
    pustaka[i].judul)
            count++
        }
    }

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

    func main() {
        var pustaka DaftarBuku
        var n int

        DaftarkanBuku(&pustaka, &n)

        CetakTerfavorit(pustaka, n)

        UrutBuku(&pustaka, n)

        Cetak5Terbaru(pustaka, n)

        var rating int
        fmt.Println("Masukkan rating buku yang ingin
    dicari:")
        fmt.Scan(&rating)
        CariBuku(pustaka, n, rating)
    }
```

Screenshoot Output

```
PS D:\praktikum alpro> go run "d:\praktikum alpro\Modul12\Unguided3.go"
Masukkan jumlah buku:
3

Masukkan data buku ke-1:
ID: 1
Judul: malamyangsunyi
Penulis: fatan
Penerbit: gramde
Eksemplar: 5
Tahun: 2020
Rating: 8
```

```
Masukkan data buku ke-2:
ID: 2
Judul: pagiyangsunyi
Penulis: azki
Penerbit: gramde
Eksemplar: 9
Tahun: 2021
Rating: 10
```

```
Masukkan data buku ke-3:
ID: 3
Judul: sepiringnasi
Penulis: kila
Penerbit: gramde
Eksemplar: 7
Tahun: 2019
Rating: 9
```

```
Buku terfavorit:
Judul: pagiyangsunyi, Penulis: azki, Penerbit: gramde, Tahun: 2021
5 Buku dengan rating tertinggi:
1. pagiyangsunyi
2. sepiringnasi
3. malamyangsunyi
Masukkan rating buku yang ingin dicari:
9
Buku ditemukan:
Judul: sepiringnasi, Penulis: kila, Penerbit: gramde, Tahun: 2019, Eksemplar: 7, Rating: 9
PS D:\praktikum alpro>
```

Deskripsi Program

Sistem manajemen perpustakaan ini mengatur data buku dengan struktur data array. Ini menyimpan informasi seperti judul, ID, penulis, penerbit, jumlah eksemplar, tahun terbit, dan rating. Untuk mengurutkan buku berdasarkan peringkat dalam urutan menurun, algoritma "insertion sort" dan algoritma "binary search" bekerja sama untuk mencari buku berdasarkan peringkat tertentu. Cara kerjanya adalah sebagai berikut: pengguna memasukkan data sejumlah buku, dan program menampilkan buku dengan rating tertinggi. Selanjutnya, data buku diurutkan dari rating tertinggi ke terendah, dan program menampilkan setidaknya lima buku dengan rating tertinggi. Terakhir, pengguna dapat mencari buku berdasarkan penilaian tertentu. Jika tidak ada yang cocok, hasilnya dapat berupa pesan bahwa buku dengan rating tersebut tidak ditemukan atau informasi tentang buku yang sesuai.