# LAPORAN PRAKTIKUM ALGORITMA DAN PEMROGRAMAN 2 MODUL XII & XIII PENGURUTAN DATA



Oleh:

ANISSA FAUZIA ISYANTI

2311102219

S1IF-11-02

**S1 TEKNIK INFORMATIKA**

**INSTITUT TEKNOLOGI TELKOM PURWOKERTO**

**2024**

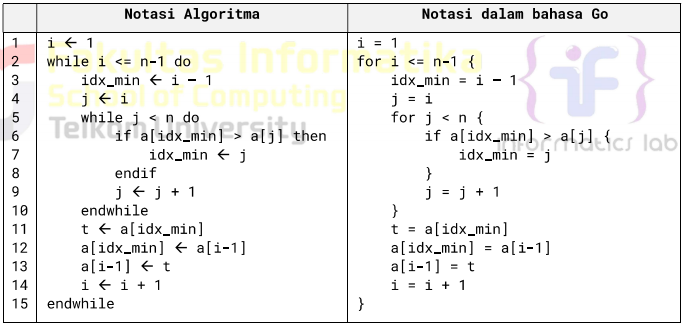
## DASAR TEORI

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”.

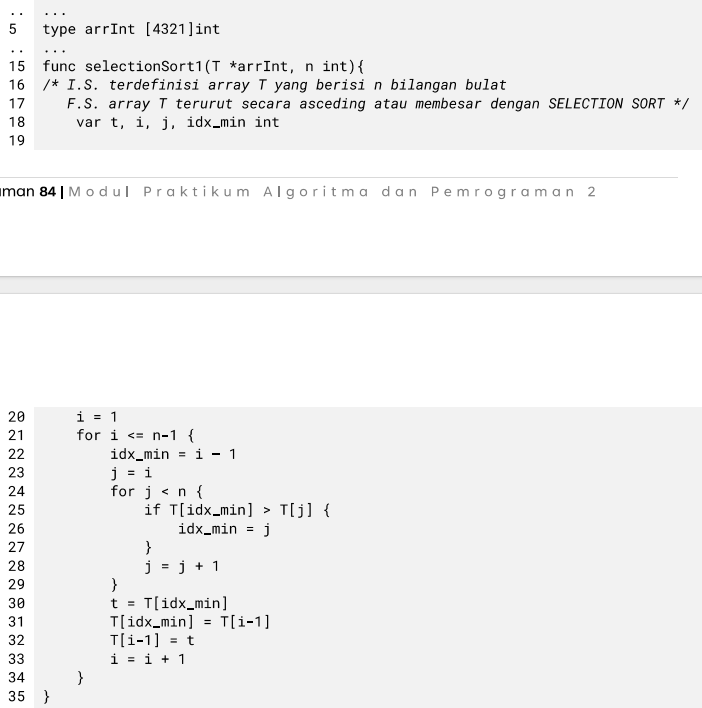
1. Cari nilai terkecil di dalam rentang data tersisa
2. Pindahkan/tukar temmpat dengan data yang berada pada posisi paling kiri pada rentang data tersisa tersebut.
3. Ulangi proses ini sampai tersisa hanya satu data saja.

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



1. Algoritma Selection Sort

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



A screenshot of a computer code

Description automatically generated

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

A screenshot of a computer code

Description automatically generated

1. 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.

1. Ulangi proses tersebut untuk setiap data yang belum erurut terhadap rangkaian data yang sudah terurut.

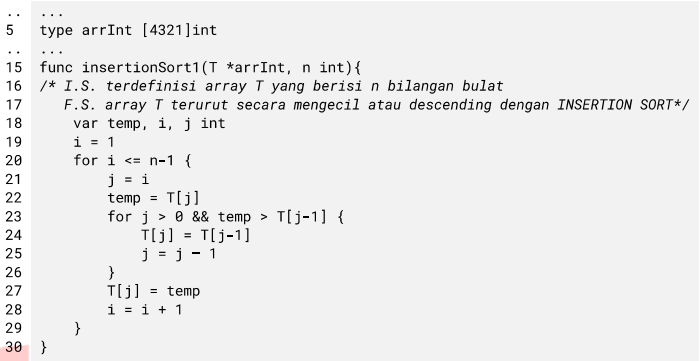
Algoritma ini dikenal dengan nama ***Insertion Sort***, yang mana pada algoritma ini melibatkan dua proses yaitu pencarian sekuensiial dan penyisipan.

A screenshot of a white sheet with black text

Description automatically generated

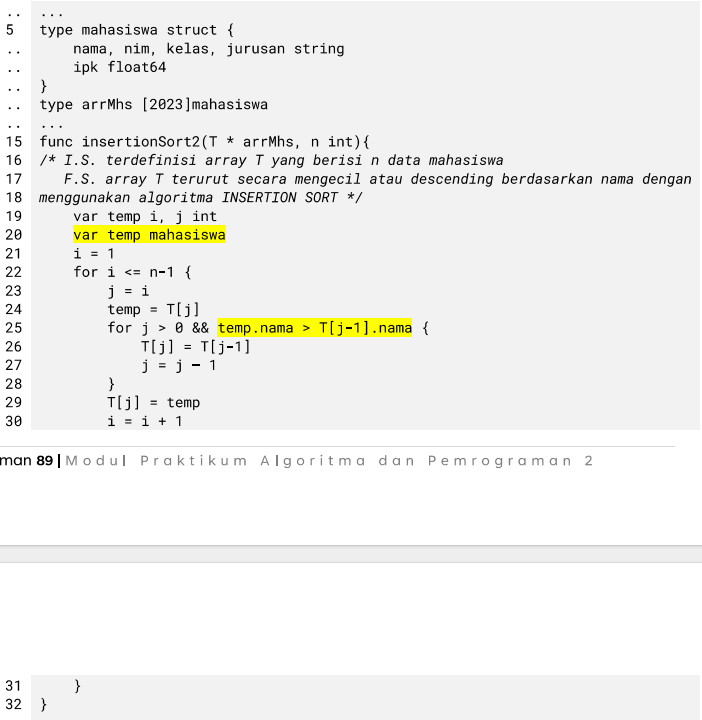
1. Algoritma Insertion Sort

Adapun algoritma *Insertion Sort* pada untuk mengurutkan array bertipe data bilangan bulat seccara mengecil atau *descending* sebagai berikut ini!



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

A screenshot of a computer

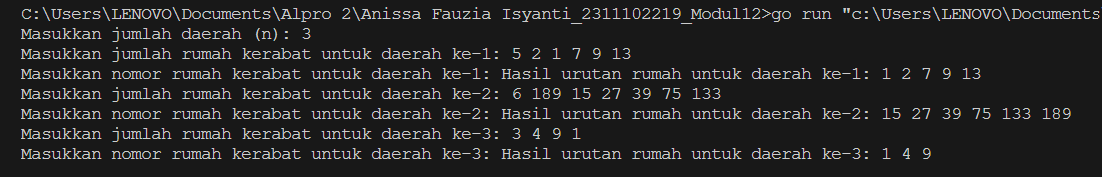
Description automatically generated

## GUIDED

### 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++ {  minIdx := i  for j := i + 1; j < n; j++ {  if arr[j] < arr[minIdx] { // Cari elemen terkecil  minIdx = j  }  }  arr[i], arr[minIdx] = arr[minIdx], 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 Program

Program ini untuk mengurutkan nomor rumah menggunakan algoritma selection sort. Pengguna diminta memasukkan jumlah daerah, kemudian untuk setiap daerah, dimasukkan jumlah rumah dan daftar nomor rumahnya. Input jumlah daerah harus lebih dari 0 dan kurang dari 1000, serta jumlah rumah lebih dari 0 dan kurang dari 1.000.000. Setelah data dimasukkan, program mengurutkan nomor rumah menggunakan selection sort dan mencetak hasilnya dalam urutan yang terurut untuk setiap daerah.

### 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 Program

A black screen with white text

Description automatically generated

A screen shot of a computer

Description automatically generated

Program ini merupakan program untuk mengurutkan bilagnan menggunakan algoritma insertion sort. Di awal program, pengguna diminta memasukkan beberapa bilangan bulat hingga bilangan negatif dimasukkan sebagai penenda akhir input. Program kemudian mengurutkan bilangan dan memeriksa apakah data memiliki selisih elemen yang tetap. Jika jarak jarak antar elemen konsisten, program akan mencetak jarak selisih. Jika tidak, program akan menccetak pesan “Data berjarak tidak tetap”.

## UNGUIDED

### Source Code

|  |
| --- |
| package main  import "fmt"  func selectionSortAsc(arr []int) {  n := len(arr)  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 selectionSortDesc(arr []int) {  n := len(arr)  for i := 0; i < n-1; i++ {  idxMax := i  for j := i + 1; j < n; j++ {  if arr[j] > arr[idxMax] {  idxMax = j  }  }  arr[i], arr[idxMax] = arr[idxMax], arr[i]  }  }  func main() {  var jmlDaerah int  fmt.Println("Input")  fmt.Scan(&jmlDaerah)  if jmlDaerah <= 0 || jmlDaerah >= 1000 {  fmt.Println("Jumlah daerah harus di antara 1 dan 999.")  return  }  data := make([][]int, jmlDaerah)  for i := 0; i < jmlDaerah; i++ {  var jmlRumah int  fmt.Scan(&jmlRumah)  if jmlRumah <= 0 || jmlRumah >= 1000000 {  fmt.Println("Jumlah rumah harus di antara 1 dan 999999.")  return  }  data[i] = make([]int, jmlRumah)  for j := 0; j < jmlRumah; j++ {  fmt.Scan(&data[i][j])  }  }  fmt.Println("\nOutput:")  for \_, daerah := range data {  var ganjil []int  var genap []int  for \_, angka := range daerah {  if angka%2 == 0 {  genap = append(genap, angka)  } else {  ganjil = append(ganjil, angka)  }  }  selectionSortAsc(ganjil)  selectionSortDesc(genap)  hasil := append(ganjil, genap...)  for \_, nilai := range hasil {  fmt.Print(nilai, " ")  }  fmt.Println()  }  } |

Screenshot Program

A screen shot of a computer

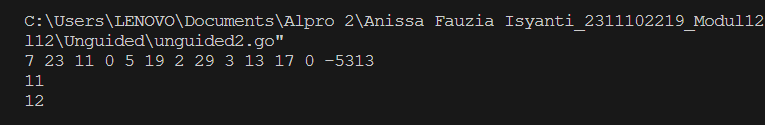
Description automatically generated

Program ini mengolah data rumah dalam beberapa daerah berdasarkan angka ganjil dan genap. Pengguna diminta memasukkan jumlah daerah (1–999) dan jumlah rumah per daerah (1–999.999), diikuti dengan data angka rumah untuk masing-masing daerah. Program memisahkan angka menjadi dua kelompok: ganjil dan genap, lalu mengurutkan angka ganjil secara ascending dan angka genap secara descending menggunakan algoritma Selection Sort. Setelah itu, kedua kelompok angka digabungkan dengan urutan ganjil terlebih dahulu, diikuti genap, dan hasilnya ditampilkan untuk setiap daerah.

### Source Code

|  |
| --- |
| package main  import "fmt"  func selectionSort(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 calculateMedian(arr []int) int {  n := len(arr)  if n%2 == 1 {  return arr[n/2]  }  return (arr[n/2-1] + arr[n/2]) / 2  }  func main() {  var input int  var data []int  for {  fmt.Scan(&input)  if input == -5313 {  break  }  if input == 0 {  if len(data) == 0 {  continue  }  selectionSort(data)  median := calculateMedian(data)  fmt.Println(median)  } else {  data = append(data, input)  }  }  } |

Screenshot Program



Program ini mengolah data angka dari input pengguna untuk menghitung nilai median. Pengguna dapat memasukkan serangkaian angka positif, di mana angka **0** digunakan untuk memicu perhitungan median dari angka-angka yang sudah dimasukkan sebelumnya, dan angka **-5313** digunakan untuk mengakhiri program. Ketika angka **0** dimasukkan, program akan mengurutkan angka yang sudah dimasukkan menggunakan algoritma Selection Sort dan menghitung median dari angka tersebut. Median dihitung sebagai angka tengah jika jumlah elemen ganjil atau rata-rata dari dua angka tengah jika jumlah elemen genap. Program ini terus menerima input hingga pengguna memasukkan **-5313**, dan akan mengabaikan input **0** jika belum ada data sebelumnya.

### Source Code

|  |
| --- |
| package main  import "fmt"  const nMax = 7919  type Buku struct {  id int  judul string  penulis string  penerbit string  eksemplar int  tahun int  rating int  }  type DaftarBuku struct {  pustaka []Buku  }  func DaftarkanBuku(pustaka \*DaftarBuku, buku Buku) {  if len(pustaka.pustaka) < nMax {  pustaka.pustaka = append(pustaka.pustaka, buku)  }  }  func CetakTerfavorit(pustaka DaftarBuku, n int) Buku {  if n == 0 {  fmt.Println("Tidak ada buku dalam pustaka.")  return Buku{}  }  terfavorit := pustaka.pustaka[0]  for i := 1; i < n; i++ {  if pustaka.pustaka[i].rating > terfavorit.rating {  terfavorit = pustaka.pustaka[i]  }  }  return terfavorit  }  func UrutBuku(pustaka \*DaftarBuku, n int) {  for i := 1; i < n; i++ {  key := pustaka.pustaka[i]  j := i - 1  for j >= 0 && pustaka.pustaka[j].rating < key.rating {  pustaka.pustaka[j+1] = pustaka.pustaka[j]  j--  }  pustaka.pustaka[j+1] = key  }  }  func Cetak5Terbaru(pustaka DaftarBuku, n int) {  if n == 0 {  fmt.Println("Tidak ada buku dalam pustaka.")  return  }  fmt.Println("5 Buku Dengan Rating Tertinggi:")  for i := 0; i < 5 && i < n; i++ {  buku := pustaka.pustaka[i]  fmt.Printf("Judul : %s\nRating : %d\n", buku.judul, buku.rating)  }  }  func CariBuku(pustaka DaftarBuku, n, r int) {  UrutBuku(&pustaka, n)  low, high := 0, n-1  for low <= high {  mid := (low + high) / 2  if pustaka.pustaka[mid].rating == r {  buku := pustaka.pustaka[mid]  fmt.Printf("Buku ditemukan:\nJudul : %s\nPenulis : %s\nPenerbit : %s\nTahun : %d\nEksemplar : %d\nRating : %d\n",  buku.judul, buku.penulis, buku.penerbit, buku.tahun, buku.eksemplar, buku.rating)  return  } else if pustaka.pustaka[mid].rating < r {  high = mid - 1  } else {  low = mid + 1  }  }  fmt.Println("Tidak ada buku dengan rating seperti itu.")  }  func main() {  var pustaka DaftarBuku  var n int  fmt.Print("Masukkan jumlah buku: ")  fmt.Scanln(&n)  for i := 0; i < n; i++ {  var id, eksemplar, tahun, rating int  var judul, penulis, penerbit string  fmt.Printf("Masukkan data buku ke-%d:\n", i+1)  fmt.Print("ID : ")  fmt.Scanln(&id)  fmt.Print("Judul : ")  fmt.Scanln(&judul)  fmt.Print("Penulis : ")  fmt.Scanln(&penulis)  fmt.Print("Penerbit : ")  fmt.Scanln(&penerbit)  fmt.Print("Eksemplar : ")  fmt.Scanln(&eksemplar)  fmt.Print("Tahun : ")  fmt.Scanln(&tahun)  fmt.Print("Rating : ")  fmt.Scanln(&rating)  buku := Buku{id, judul, penulis, penerbit, eksemplar, tahun, rating}  DaftarkanBuku(&pustaka, buku)  fmt.Println()  }  terfavorit := CetakTerfavorit(pustaka, n)  fmt.Println("Buku Terfavorit:")  fmt.Printf("Judul : %s\nPenulis : %s\nPenerbit : %s\nTahun : %d\nRating : %d\n",  terfavorit.judul, terfavorit.penulis, terfavorit.penerbit, terfavorit.tahun, terfavorit.rating)  UrutBuku(&pustaka, n)  Cetak5Terbaru(pustaka, n)  var targetRating int  fmt.Print("\nMasukkan rating buku yang ingin dicari: ")  fmt.Scanln(&targetRating)  CariBuku(pustaka, n, targetRating)  } |

Screenshot Program

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Program pengelolaan pustaka buku yang memungkinkan pengguna untuk memasukkan data buku, mengurutkan buku berdasarkan rating, dan melakukan pencarian buku berdasarkan rating tertentu. Program ini mendefinisikan dua tipe data utama, yaitu Buku yang berisi informasi terkait buku seperti ID, judul, penulis, penerbit, jumlah eksemplar, tahun terbit, dan rating; serta DaftarBuku yang merupakan koleksi dari buku-buku yang terdaftar. Pengguna dapat memasukkan sejumlah buku ke dalam pustaka, di mana data buku dapat berupa judul, penulis, penerbit, eksemplar, tahun terbit, dan rating. Program ini menyediakan berbagai fungsi: DaftarkanBuku untuk menambahkan buku ke pustaka, CetakTerfavorit untuk menampilkan buku dengan rating tertinggi, UrutBuku untuk mengurutkan buku berdasarkan rating tertinggi, Cetak5Terbaru untuk mencetak lima buku dengan rating tertinggi, dan CariBuku untuk mencari buku berdasarkan rating menggunakan pencarian biner. Program ini juga menggunakan bufio.Scanner untuk menangani input string yang lebih baik.