**LAPORAN PRAKTIKUM ALGORITMA DAN PEMROGRAMAN 2**

**MODUL 12**

**PENGURUTAN DATA**



Disusun Oleh:

**NAMA : IKRAM IRIANSYAH**

**NIM : 2311102184**

**KELAS : IF-11-02**

**S1 TEKNIK INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**

**2024**

# DASAR TEORI

# Selection Sort

Pengertian:

Selection Sort adalah algoritma pengurutan yang bekerja dengan cara mencari elemen terkecil (atau terbesar, tergantung urutan) dari array yang belum terurut dan menukarnya dengan elemen pertama dari array tersebut. Proses ini diulang untuk elemen berikutnya hingga seluruh array terurut.

Cara kerja:

* 1. Mulai dari elemen pertama, cari elemen terkecil di antara elemen yang belum terurut.
  2. Tukar elemen terkecil tersebut dengan elemen pertama.
  3. Pindah ke elemen berikutnya dan ulangi proses hingga semua elemen terurut.

# Insertion Sort

Pengertian:

Insertion Sort adalah algoritma pengurutan yang membangun subarray terurut satu per satu. Algoritma ini mengambil elemen dari bagian array yang belum diurutkan dan menyisipkannya ke posisi yang tepat dalam subarray yang sudah terurut.

Cara kerja:

1. Anggap elemen pertama sudah terurut.
2. Ambil elemen berikutnya (key) dari bagian yang belum diurutkan.
3. Bandingkan key dengan elemen-elemen di subarray terurut dari belakang ke depan.
4. Geser elemen-elemen yang lebih besar satu posisi ke kanan.
5. Sisipkan key pada posisi yang tepat.
6. Ulangi proses hingga seluruh array terurut.

# GUIDED

**GUIDED 1**

|  |
| --- |
| 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]%2 == 1 {                  if arr[j] < arr[maxIdx] { // Cari elemen terbesar                      maxIdx = j                  }              } else {                  if arr[j] > arr[maxIdx] {                      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**

A black screen with white text

Description automatically generated

**DESKRIPSI PROGRAM**

Program ini membaca sejumlah data nomor rumah kerabat di beberapa daerah, mengurutkannya, dan menampilkan hasilnya. Untuk setiap daerah, pengguna memasukkan jumlah rumah kerabat dan nomor rumah masing-masing. Program menggunakan algoritma **selection sort** untuk mengurutkan nomor rumah secara menurun (descending). Hasil pengurutan untuk setiap daerah ditampilkan dalam urutan terbesar ke terkecil.

**GUIDED 2**

|  |
| --- |
| 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")      }  } |

**OUTPUT**

**A black screen with white text

Description automatically generated**

**DESKRIPSI PROGRAM**

Program ini membaca sejumlah data bilangan bulat dari pengguna hingga bilangan negatif dimasukkan sebagai penanda akhir input. Data tersebut diurutkan secara menaik menggunakan algoritma **insertion sort**. Setelah data diurutkan, program memeriksa apakah selisih antara elemen-elemen dalam array memiliki jarak yang tetap. Jika data berjarak tetap, program menampilkan selisihnya jika tidak, program menyatakan bahwa data tidak berjarak tetap. Output program mencakup hasil pengurutan data serta informasi tentang konsistensi jarak antar elemen.

# UNGUIDED

# UNGUIDED 1

|  |
| --- |
| //IKRAM IRIANSYAH//2311102184package mainimport "fmt"func slice\_data(arr []int) ([]int, []int) {    var ganjil, genap []int    for \_, num := range arr {        if num%2 == 0 {            genap = append(genap, num)        } else {            ganjil = append(ganjil, num)        }    }    return ganjil, genap}func asc(arr []int) {    n := len(arr)    for i := 0; i < n-1; i++ {        for j := 0; j < n-i-1; j++ {            if arr[j] > arr[j+1] {                arr[j], arr[j+1] = arr[j+1], arr[j]            }        }    }}func desc(arr []int) {    n := len(arr)    for i := 0; i < n-1; i++ {        for j := 0; j < n-i-1; j++ {            if arr[j] < arr[j+1] {                arr[j], arr[j+1] = arr[j+1], arr[j]            }        }    }}func main() {    var n int    fmt.Print("Masukkan Jumlah Daerah (n) : ")    fmt.Scan(&n)    for i := 0; i < n; i++ {        fmt.Printf("\nMasukkan Jumlah Rumah Daerah %d : ", i+1)        var m int        fmt.Scan(&m)        arr := make([]int, m)        fmt.Printf("Masukkan %d Nomor Rumah Daerah %d : ", m, i+1)        for j := 0; j < m; j++ {            fmt.Scan(&arr[j])        }        ganjil, genap := slice\_data(arr)        desc(genap)        asc(ganjil)        fmt.Printf("Nomor Rumah Terurut [ Ganjil(+), Genap(-) ] Daerah %d : ", i+1)        for \_, num := range ganjil {            fmt.Printf("%d ", num)        }        for \_, num := range genap {            fmt.Printf("%d ", num)        }        fmt.Println()    }} |

# OUTPUT

# A computer screen with text Description automatically generated

**DESKRIPSI PROGRAM**

Program ini memisahkan nomor rumah di setiap daerah ke dalam dua kelompok berdasarkan paritasnya: ganjil dan genap. Fungsi **slice\_data** memisahkan elemen array ke dalam dua array terpisah untuk ganjil dan genap. Fungsi **asc** dan **desc** mengurutkan array dalam urutan menaik dan menurun menggunakan algoritma bubble sort. Program meminta pengguna memasukkan jumlah daerah, nomor rumah di setiap daerah, lalu menampilkan nomor ganjil dalam urutan menaik diikuti nomor genap dalam urutan menurun untuk setiap daerah.

**UNGUIDED 2**

|  |
| --- |
| //IKRAM IRIANSYAH  //2311102184  package main  import "fmt"  func sort(data []int) {    n := len(data)    for i := 0; i < n-1; i++ {      minIdx := i      for j := i + 1; j < n; j++ {        if data[j] < data[minIdx] {          minIdx = j        }      }      // Tukar elemen      data[i], data[minIdx] = data[minIdx], data[i]    }  }  func median(data []int) int {    n := len(data)    if n%2 == 0 {      return (data[n/2-1] + data[n/2]) / 2    }    return data[n/2]  }  func main() {    var input int    var data []int    var groups [][]int    fmt.Println("Masukkan Data (akhiri dengan -5313) ")    for {      fmt.Scan(&input)      if input == -5313 {        break      }      if input == 0 {        groups = append(groups, append([]int{}, data...))      } else {        data = append(data, input)      }    }    for i, group := range groups {      sort(group)      median := median(group)      fmt.Printf("Median %d : %d\n", i+1, median)    }  } |

**OUTPUT**

A black screen with colorful text

Description automatically generated

**DESKRIPSI PROGRAM**

Program ini mengelompokkan data integer ke dalam beberapa grup, mengurutkannya menggunakan algoritma **selection sort**, dan menghitung median dari setiap grup. Fungsi sort mengurutkan elemen grup, sementara fungsi **median** menghitung nilai tengah grup berdasarkan panjangnya (genap atau ganjil). Pengguna memasukkan data integer yang dipisahkan dengan 0 untuk menandai grup baru, dan **-5313** untuk mengakhiri input. Program menampilkan median dari setiap grup yang telah diurutkan.

**UNGUIDED 3**

|  |
| --- |
| //IKRAM IRIANSYAH  //2311102184  package main  import (      "fmt"  )  type Buku struct {      ID        string      Judul     string      Penulis   string      Penerbit  string      Eksemplar int      Tahun     int      Rating    int  }  type DaftarBuku struct {      Pustaka  []Buku      NPustaka int  }  func TambahBuku(pustaka \*DaftarBuku) {      var n int      fmt.Print("Masukkan Jumlah Buku : ")      fmt.Scan(&n)      pustaka.NPustaka = n      pustaka.Pustaka = make([]Buku, n)      for i := 0; i < n; i++ {          fmt.Printf("\nMasukkan Data Buku %d! \n", i+1)          fmt.Print("ID : ")          fmt.Scan(&pustaka.Pustaka[i].ID)          fmt.Print("Judul : ")          fmt.Scan(&pustaka.Pustaka[i].Judul)          fmt.Print("Penulis : ")          fmt.Scan(&pustaka.Pustaka[i].Penulis)          fmt.Print("Penerbit : ")          fmt.Scan(&pustaka.Pustaka[i].Penerbit)          fmt.Print("Eksemplar : ")          fmt.Scan(&pustaka.Pustaka[i].Eksemplar)          fmt.Print("Tahun : ")          fmt.Scan(&pustaka.Pustaka[i].Tahun)          fmt.Print("Rating : ")          fmt.Scan(&pustaka.Pustaka[i].Rating)      }  }  func CetakBuku(pustaka DaftarBuku) {      fmt.Println("\n= DATA BUKU = ")      fmt.Printf("%-5s %-25s %-20s %-20s %-10s %-10s %-5s\n", "ID", "JUDUL", "PENULIS", "PENERBIT", "EKSEMPLAR", "TAHUN", "RATING")      for \_, buku := range pustaka.Pustaka {          fmt.Printf("%-5s %-25s %-20s %-20s %-10d %-10d %-5d\n",              buku.ID, buku.Judul, buku.Penulis, buku.Penerbit, buku.Eksemplar, buku.Tahun, buku.Rating)      }  }  func UrutkanBuku(pustaka \*DaftarBuku) {      n := pustaka.NPustaka      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 CetakBukuTeratas(pustaka DaftarBuku) {      fmt.Println("\n= 5 BUKU TERATAS BERDASARKAN RATING = ")      fmt.Printf("%-25s %-5s\n", "JUDUL", "RATING")      for i := 0; i < 5 && i < pustaka.NPustaka; i++ {          fmt.Printf("%-25s %-5d\n", pustaka.Pustaka[i].Judul, pustaka.Pustaka[i].Rating)      }  }  func CariBuku(pustaka DaftarBuku) {      var rating int      fmt.Print("\nMasukan Rating :")      fmt.Scan(&rating)      left, right := 0, pustaka.NPustaka-1      found := false      for left <= right {          mid := (left + right) / 2          if pustaka.Pustaka[mid].Rating == rating {              found = true              buku := pustaka.Pustaka[mid]              fmt.Printf("\n= BUKU DENGAN RATING %d = \n", rating)              fmt.Printf("%-5s %-25s %-20s %-20s %-10s %-10s %-5s\n", "ID", "JUDUL", "PENULIS", "PENERBIT", "EKSEMPLAR", "TAHUN", "RATING")              fmt.Printf("%-5s %-25s %-20s %-20s %-10d %-10d %-5d\n",                  buku.ID, buku.Judul, buku.Penulis, buku.Penerbit, buku.Eksemplar, buku.Tahun, buku.Rating)              break          } else if pustaka.Pustaka[mid].Rating < rating {              right = mid - 1          } else {              left = mid + 1          }      }      if !found {          fmt.Println("\nTidak ada buku dengan rating tersebut.")      }  }  func main() {      var pustaka DaftarBuku      TambahBuku(&pustaka)      UrutkanBuku(&pustaka)      CetakBuku(pustaka)      CetakBukuTeratas(pustaka)      CariBuku(pustaka)  } |

**OUTPUT**

A black screen with white text

Description automatically generated

A black background with white text

Description automatically generated

A black screen with white text

Description automatically generated

**DESKRIPSI PROGRAM**

Program ini mengelola data buku dengan menyediakan fitur untuk menambah, mencetak, mengurutkan, dan mencari data buku berdasarkan rating. Fungsi TambahBuku meminta pengguna memasukkan data buku, sementara fungsi UrutkanBuku mengurutkan daftar buku berdasarkan rating menggunakan insertion sort. Fungsi CetakBuku mencetak semua buku, sedangkan CetakBukuTeratas menampilkan 5 buku dengan rating tertinggi. Fungsi CariBuku menggunakan pencarian biner untuk menemukan buku dengan rating tertentu.