LAPORAN PRAKTIKUM

ALGORITMA DAN PEMROGRAMAN 2

MODUL 8

“PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA”

DISUSUN OLEH:

RAIHAN ADI ARBA

103112400071

S1 IF-12-01

DOSEN:

Dimas Fanny Hebrasianto Permadi, S.ST., M.Kom

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024/2025

DASAR TEORI

Algoritma pencarian nilai minimum dan maksimum adalah metode untuk menemukan nilai terkecil dan terbesar dalam suatu kumpulan data. Salah satu pendekatan yang umum digunakan adalah pencarian linear, di mana setiap elemen data diperiksa satu per satu secara berurutan. Proses ini dimulai dengan menginisialisasi nilai minimum dan maksimum menggunakan elemen pertama array atau slice. Selanjutnya, algoritma melakukan iterasi melalui sisa elemen, membandingkan setiap nilai dengan nilai minimum dan maksimum sementara. Jika ditemukan elemen yang lebih kecil dari nilai minimum saat ini, nilai minimum diperbarui. Demikian pula, jika ada elemen yang lebih besar dari nilai maksimum sementara, nilai maksimum akan disesuaikan. Setelah seluruh data diperiksa, nilai minimum dan maksimum yang diperoleh merupakan hasil akhir pencarian.

Algoritma ini memiliki keunggulan dalam kesederhanaan dan kemampuannya untuk diterapkan dalam berbagai konteks. Misalnya, dalam pengolahan data seperti pengukuran berat badan balita, pertumbuhan tinggi tanaman, fluktuasi harga buku, atau analisis nilai ujian siswa, pencarian nilai ekstrim sangat penting untuk mendukung proses analisis dan pengambilan keputusan. Dengan memeriksa setiap elemen secara sistematis, metode ini menjamin akurasi hasil meskipun data yang diolah berukuran besar atau tidak terurut.

A. GUIDED

1. Source code :

```go
package main

import (
    "fmt"
    "sort"
)

func main() {
    // Array awal
    data := []float64{2, 7, 9, 1, 5, 6, 18, 13, 25, 20}
    target := 13.0

    fmt.Println("Sequential Search (data tidak perlu urut):")
    idxSeq, iterSeq := sequentialSearch(data, target)
    if idxSeq != -1 {
        fmt.Printf("Hasil: Ditemukan di indeks %d dalam %d
langkah\n\n", idxSeq, iterSeq)
    } else {
        fmt.Printf("Hasil: Tidak ditemukan setelah %d langkah\n\n",
iterSeq)
    }

    // Binary search perlu array diurutkan
    sort.Float64s(data)
    fmt.Println("Binary Search (setelah data diurutkan):")
    fmt.Println("Data terurut:", data)

    idxBin, iterBin := binarySearch(data, target)
    if idxBin != -1 {
        fmt.Printf("Hasil: Ditemukan di indeks %d dalam %d langkah\n",
idxBin, iterBin)
    } else {
        fmt.Printf("Hasil: Tidak ditemukan setelah %d langkah\n",
iterBin)
    }
}

func sequentialSearch(arr []float64, target float64) (int, int) {
    iterations := 0
    for i, val := range arr {
        iterations++
        fmt.Printf("Sequential step %d: cek arr[%d] = %.1f\n",
iterations, i, val)
        if val == target {
            return i, iterations
        }
```

```
    }
    return -1, iterations
}

func binarySearch(arr []float64, target float64) (int, int) {
    iterations := 0
    low := 0
    high := len(arr) - 1

    for low <= high {
        iterations++
        mid := (low + high) / 2
        fmt.Printf("Binary step %d: cek arr[%d] = %.1f\n", iterations,
mid, arr[mid])

        if arr[mid] == target {
            return mid, iterations
        } else if target < arr[mid] {
            high = mid -1
        } else {
            low = mid + 1
        }
    }
    return -1, iterations
}
```

Output :

```
raihan@Raihans-MacBook-Pro Laprak-Modul-8 % go run "/Users/raihan/Documer
tHub/PRAKTIKUM-2/Laprak-Modul-8/103112400071_MODUL8/tempCodeRunnerFile.go
Sequential Search (data tidak perlu urut):
Sequential step 1: cek arr[0] = 2.0
Sequential step 2: cek arr[1] = 7.0
Sequential step 3: cek arr[2] = 9.0
Sequential step 4: cek arr[3] = 1.0
Sequential step 5: cek arr[4] = 5.0
Sequential step 6: cek arr[5] = 6.0
Sequential step 7: cek arr[6] = 18.0
Sequential step 8: cek arr[7] = 13.0
Hasil: Ditemukan di indeks 7 dalam 8 langkah

Binary Search (setelah data diurutkan):
Data terurut: [1 2 5 6 7 9 13 18 20 25]
Binary step 1: cek arr[4] = 7.0
Binary step 2: cek arr[7] = 18.0
Binary step 3: cek arr[5] = 9.0
Binary step 4: cek arr[6] = 13.0
Hasil: Ditemukan di indeks 6 dalam 4 langkah
```

Deskripsi :
Program ini, ditulis dalam bahasa Go, mengimplementasikan dua algoritma pencarian, yaitu Sequential Search dan Binary Search, untuk mencari nilai target 13.0 dalam array float64. Sequential Search memeriksa setiap elemen array yang tidak diurutkan secara berurutan, menemukan target di indeks 7 setelah 8 langkah. Sebaliknya, Binary Search memerlukan array yang telah diurutkan terlebih dahulu, kemudian menggunakan pendekatan pembagian dua untuk menemukan target di indeks 6 hanya dalam 4 langkah. Dengan menampilkan setiap langkah dan jumlah iterasi, program ini mendemonstrasikan bahwa Binary Search jauh lebih efisien dibandingkan Sequential Search untuk data yang telah diurutkan, meskipun memerlukan langkah pengurutan awal.

2. Source code :

```go
package main

import (
    "fmt"
    "sort"
)

type mahasiswa struct {
    nama, nim, kelas, jurusan string
    ipk                       float64
}

type arrMhs [2023]mahasiswa

// Sequential Search berdasarkan nama
func SeqSearch_3(T arrMhs, n int, X string) (int, int) {
    var found int = -1
    var j int = 0
    var iterasi int = 0

    for j < n && found == -1 {
        iterasi++
        if T[j].nama == X {
            found = j
        }
        j++
    }
    return found, iterasi
}

// Binary Search berdasarkan NIM (data harus sudah terurut berdasarkan
nim)
func BinarySearch_3(T arrMhs, n int, X string) (int, int) {
    var found int = -1
    var med int
```

```
    var kr int = 0
    var kn int = n - 1
    var iterasi int = 0

    for kr <= kn && found == -1 {
        iterasi++
        med = (kr + kn) / 2
        if X < T[med].nim {
            kn = med - 1
        } else if X > T[med].nim {
            kr = med + 1
        } else {
            found = med
        }
    }
    return found, iterasi
}
func main() {
    var data arrMhs
    n := 10

    // Mengisi data secara manual
    data = arrMhs{
        {nama: "Ari", nim: "2201", kelas: "A", jurusan: "Informatika",
ipk: 3.4},
        {nama: "Budi", nim: "2203", kelas: "A", jurusan: "Informatika",
ipk: 3.6},
        {nama: "Cici", nim: "2202", kelas: "B", jurusan: "Sistem
Informasi", ipk: 3.5},
        {nama: "Dina", nim: "2205", kelas: "A", jurusan: "Informatika",
ipk: 3.3},
        {nama: "Eko", nim: "2204", kelas: "B", jurusan: "Sistem
Informasi", ipk: 3.7},
        {nama: "Fajar", nim: "2206", kelas: "C", jurusan: "Informatika",
ipk: 3.1},
        {nama: "Gita", nim: "2209", kelas: "C", jurusan: "Informatika",
ipk: 3.8},
        {nama: "Hana", nim: "2208", kelas: "B", jurusan: "Sistem
Informasi", ipk: 3.2},
        {nama: "Iwan", nim: "2207", kelas: "C", jurusan: "Informatika",
ipk: 3.0},
        {nama: "Joko", nim: "2210", kelas: "A", jurusan: "Informatika",
ipk: 3.9},
    }

    // Pencarian Sequential Search berdasarkan nama
    namaDicari := "Fajar"
    idxSeq, iterSeq := SeqSearch_3(data, n, namaDicari)

    fmt.Printf("Sequential Search - Cari nama '%s'\n", namaDicari)
    if idxSeq != -1 {
```

```
        fmt.Printf("Ditemukan di indeks: %d, Iterasi: %d\n", idxSeq,
iterSeq)
    } else {
        fmt.Printf("Tidak ditemukan, Iterasi: %d\n", iterSeq)
    }

    // Urutkan data berdasarkan NIM untuk binary search
    sort.Slice(data[:n], func(i, j int) bool {
        return data[i].nim < data[j].nim
    })

    // Pencarian Binary Search berdasarkan NIM
    nimDicari := "2206"
    idxBin, iterBin := BinarySearch_3(data, n, nimDicari)

    fmt.Printf("\nBinary Search - Cari NIM '%s'\n", nimDicari)
    if idxBin != -1 {
        fmt.Printf("Ditemukan di indeks: %d, Iterasi: %d\n", idxBin,
iterBin)
    } else {
        fmt.Printf("Tidak ditemukan, Iterasi: %d\n", iterBin)
    }
}
```

Output :

```
raihan@Raihans-MacBook-Pro Laprak-Modul-8 % go run "/Users/raihan/Documents/Gi
tHub/PRAKTIKUM-2/Laprak-Modul-8/103112400071_MODUL8/103112400071_Guided2.go"
Sequential Search - Cari nama 'Fajar'
Ditemukan di indeks: 5, Iterasi: 6

Binary Search - Cari NIM '2206'
Ditemukan di indeks: 5, Iterasi: 3
```

Deskripsi :

Program ini, ditulis dalam bahasa Go, mengimplementasikan Sequential Search dan Binary Search untuk mencari data mahasiswa dalam array struct berdasarkan nama dan NIM. Sequential Search mencari nama "Fajar" dalam array yang tidak diurutkan, menemukannya di indeks 5 setelah 6 iterasi. Binary Search mencari NIM "2206" dalam array yang telah diurutkan berdasarkan NIM, menemukannya di indeks 5 dalam 3 iterasi. Program menunjukkan bahwa Binary Search lebih efisien dengan iterasi lebih sedikit, tetapi memerlukan data yang sudah terurut. Data mahasiswa mencakup nama, NIM, kelas, jurusan, dan IPK, yang diisi secara manual untuk 10 entri.

B. UNGUIDED

1. Latihan 1

Source Code:

```
// Raihan Adi Arba
// 103112400071

package main

import (
    "fmt"
)

const jumlahCalon = 20

func inisialisasiCalon() []int {
    calon := make([]int, jumlahCalon)
    for i := 0; i < jumlahCalon; i++ {
        calon[i] = i + 1
    }
    return calon
}

func hitungSuara(calon []int) (totalMasuk int, totalSah int, jumlahVote
[]int) {
    jumlahVote = make([]int, jumlahCalon)
    var suara int

    for {
        _, err := fmt.Scan(&suara)
        if err != nil || suara == 0 {
            break
        }

        totalMasuk++

        for i := 0; i < jumlahCalon; i++ {
            if suara == calon[i] {
                jumlahVote[i]++
                totalSah++
                break
            }
        }
    }

    return totalMasuk, totalSah, jumlahVote
}
```

```go
func tampilkanHasil(totalMasuk, totalSah int, calon []int, jumlahVote
[]int) {
    fmt.Println("Suara masuk:", totalMasuk)
    fmt.Println("Suara sah:", totalSah)
    for i := 0; i < jumlahCalon; i++ {
        if jumlahVote[i] > 0 {
            fmt.Printf("%d: %d\n", calon[i], jumlahVote[i])
        }
    }
}

func main() {
    calon := inisialisasiCalon()
    totalMasuk, totalSah, jumlahVote := hitungSuara(calon)
    tampilkanHasil(totalMasuk, totalSah, calon, jumlahVote)
}
```

Output :

```
raihan@Raihans-MacBook-Pro Laprak-Modul-8 % go run "/Users/raihan/Documents/Gi
tHub/PRAKTIKUM-2/Laprak-Modul-8/103112400071_MODUL8/103112400071_Unguided1.go"
292
2
2
2
22

3
2
23
2
32
2
0
Suara masuk: 11
Suara sah: 7
2: 6
3: 1
```

Penjelasan Program:

Program ini, mengimplementasikan sistem penghitungan suara untuk 20 calon. Fungsi inisialisasiCalon membuat array nomor calon (1 hingga 20). Fungsi hitungSuara menerima input suara secara berulang hingga input 0 atau tidak valid, menghitung total suara masuk, suara sah, dan jumlah vote per calon. Fungsi tampilkanHasil menampilkan hasil, termasuk total suara masuk, suara sah, dan jumlah vote untuk calon yang mendapat suara. Dalam contoh eksekusi, dari 11 suara masuk, 7 suara sah, dengan calon nomor 2 mendapat 6 suara dan calon nomor 3 mendapat 1 suara.

2. Latihan 2

Source Code:

```go
// Raihan Adi Arba
// 103112400071

package main

import "fmt"

func main() {
    const N = 20
    var (
        suara      [N + 1]int
        input      int
        totalSuara int
        suaraSah   int
        ketua      int
        wakil      int
    )

    for {
        fmt.Scan(&input)
        if input == 0 {
            break
        }
        totalSuara++

        if input >= 1 && input <= N {
            suara[input]++
            suaraSah++
        }
    }

    fmt.Printf("Suara masuk: %d\n", totalSuara)
    fmt.Printf("Suara sah: %d\n", suaraSah)

    for i := 1; i <= N; i++ {
        if suara[i] > suara[ketua] || (suara[i] == suara[ketua] && i < ketua) {
            wakil = ketua
            ketua = i
        } else if suara[i] > suara[wakil] || (suara[i] == suara[wakil] && i < wakil && i != ketua) {
            wakil = i
        }
    }

    fmt.Printf("Ketua RT: %d\n", ketua)
    fmt.Printf("Wakil ketua: %d\n", wakil)
}
```

Output:

```
raihan@Raihans-MacBook-Pro Laprak-Modul-8 % go run "/Users/raihan/Documents/GitHub/PRAKTIKUM-2/Laprak-Modul-8/103112400071_MODUL8/103112400071_Unguided2.go"

28
292
2
3
2
0
Suara masuk: 5
Suara sah: 3
Ketua RT: 2
Wakil ketua: 3
```

Deskripsi Program:

Program ini mengimplementasikan sistem penghitungan suara untuk pemilihan Ketua dan Wakil Ketua RT dengan maksimal 20 calon. Program menerima input suara secara berulang hingga input 0, menghitung total suara masuk dan suara sah (input antara 1 hingga 20). Suara untuk setiap calon disimpan dalam array, lalu program menentukan Ketua (calon dengan suara terbanyak, atau nomor terkecil jika seri) dan Wakil Ketua (calon dengan suara terbanyak kedua, atau nomor terkecil jika seri). Dalam contoh eksekusi, dari 5 suara masuk, 3 suara sah, calon nomor 2 menjadi Ketua RT dan calon nomor 3 menjadi Wakil Ketua.

3. Latihan 3

Source Code:

```go
// Raihan Adi Arba
// 103112400071

package main

import "fmt"

const NMAX = 1000000

var data [NMAX]int

func main() {
    var n, k int
    fmt.Scan(&n, &k)

    isiArray(n)
    idx := posisi(n, k)

    if idx == -1 {
        fmt.Println("TIDAK ADA")
```

```
    } else {
        fmt.Println(idx)
    }
}

func isiArray(n int) {
    for i := 0; i < n; i++ {
        fmt.Scan(&data[i])
    }
}

func posisi(n, k int) int {
    kiri, kanan := 0, n-1
    for kiri <= kanan {
        tengah := (kiri + kanan) / 2
        switch {
        case data[tengah] == k:
            return tengah
        case data[tengah] < k:
            kiri = tengah + 1
        default:
            kanan = tengah - 1
        }
    }
    return -1
}
```

Output:

```
raihan@Raihans-MacBook-Pro Laprak-Modul-8 % go run "/Users/raihan/Documents/GitHub/PRAKTIKUM-2/Laprak-Modul-8/103112400071_MODUL8/103112400071_Unguided3.go"

5 23
10 15 12 13 20 25
TIDAK ADA
raihan@Raihans-MacBook-Pro Laprak-Modul-8 % 25
zsh: command not found: 25
raihan@Raihans-MacBook-Pro Laprak-Modul-8 % go run "/Users/raihan/Documents/GitHub/PRAKTIKUM-2/Laprak-Modul-8/103112400071_MODUL8/103112400071_Unguided3.go"

5 23
10 15 23 40 50
2
raihan@Raihans-MacBook-Pro Laprak-Modul-8 %
```

Deskripsi Program:

Program ini mengimplementasikan Binary Search untuk mencari nilai k dalam array berukuran maksimal 1.000.000. Fungsi isiArray mengisi array dengan n elemen yang diinput pengguna, dan fungsi posisi menggunakan Binary Search untuk menemukan indeks nilai k dalam array yang diasumsikan terurut. Jika k ditemukan, program mengembalikan indeksnya; jika tidak, mengembalikan -1 dan mencetak "TIDAK

ADA". Dalam eksekusi pertama, dengan input n=5, k=23, dan array [10, 15, 12, 13, 20], hasilnya "TIDAK ADA" karena 23 tidak ditemukan. Pada eksekusi kedua, dengan array [10, 15, 23, 40, 50], 23 ditemukan di indeks 2, sehingga outputnya adalah 2.

DAFTAR PUSTAKA

Prayogo, N. A. (2021). ***Dasar Pemrograman Go. Ebook***