LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL 11

PENCARIAN NILAI ACAK PADA HIMPUNAN DATA

Oleh:

M.HANIF AL FAIZ

103112400042

12IF-01

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

I. DASAR TEORI

Pemrograman dengan bahasa Go (Golang) menekankan prinsip kesederhanaan, efisiensi, dan dukungan konkurensi, dengan fitur seperti penentuan tipe data saat kompilasi, pengelolaan memori otomatis melalui *garbage collection*, serta kemampuan mengembalikan multi-nilai dari fungsi. Dalam konteks pencarian data, dua algoritma utama yang dibahas adalah Sequential Search dan Binary Search. Sequential Search bekerja dengan memeriksa setiap elemen secara berurutan, cocok untuk data tidak terurut namun memiliki kompleksitas waktu O(n). Sementara itu, Binary Search memerlukan data terurut dan membagi interval pencarian secara rekursif, mencapai efisiensi O(log n). Kedua metode ini diimplementasikan dalam Go menggunakan struktur dasar seperti array, slice, serta kontrol alur seperti perulangan dan percabangan, sementara validasi input digunakan untuk memastikan keandalan program.

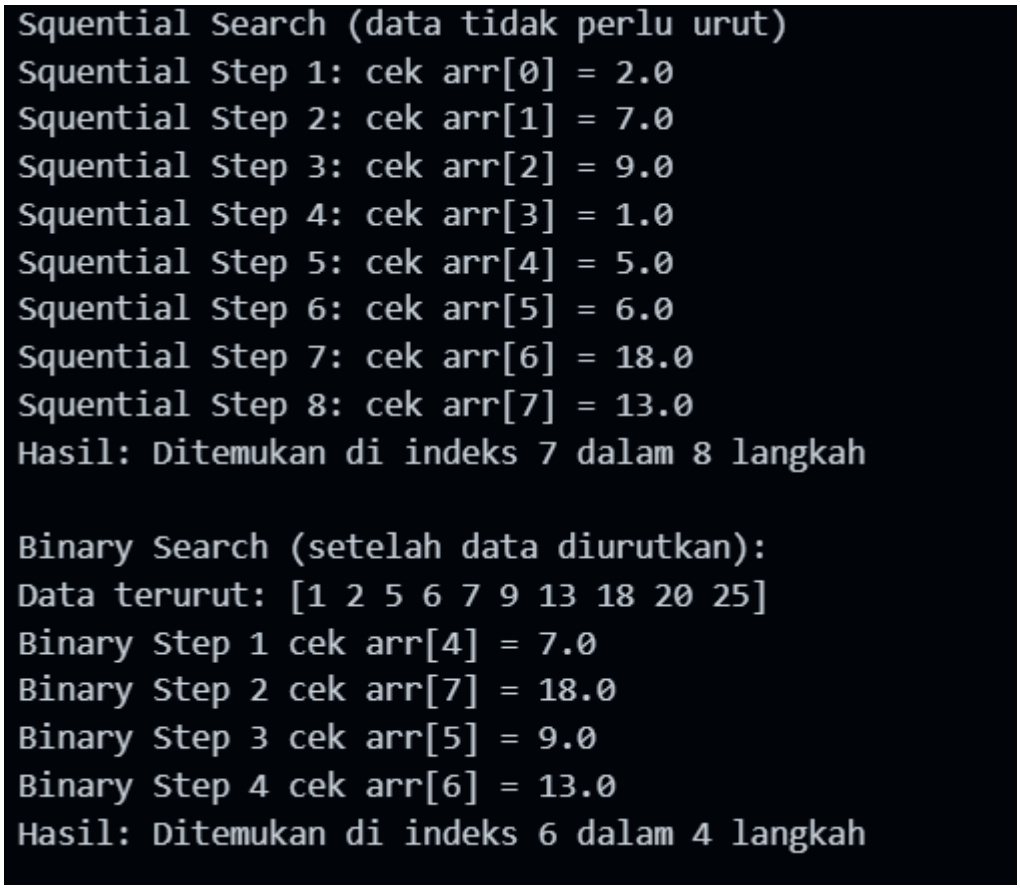
II. GUIDED

1

```
package main

import (
    "fmt"
    "sort"
)

func sequentialSearch(arr []float64, target float64) (int, int) {
    iterations := 0
    for i, val := range arr {
        iterations++
        fmt.Printf("Squential 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
}

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

    fmt.Println("Squential 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)
    }

    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)
    }
}
```

OUTPUT:



DESKRIPSI: aplikasi sederhana dalam bahasa Go yang menghitung total gaji karyawan

2.

```
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:

```
Sequential Search - Cari nama 'Fajar'
Ditemukan di indeks: 5, Iterasi: 6

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

Deskripsi:

aplikasi dalam bahasa Go yang mendemonstrasikan dua metode pencarian data mahasiswa:

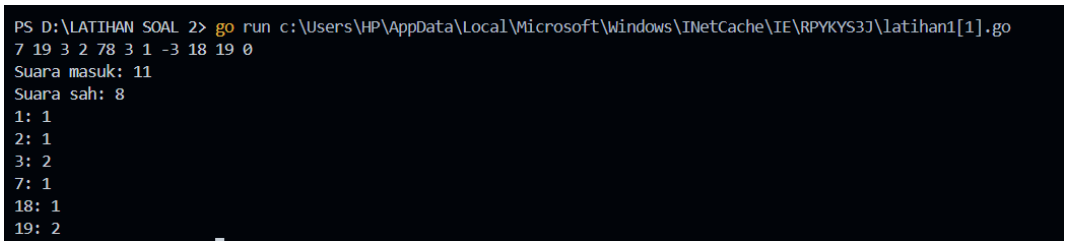
III. UNGUIDED

1

```
// M.HANIF AL FAIZ
// NIM: 103112400042
package main

import (
    "bufio"
    "fmt"
    "os"
    "sort"
    "strconv"
    "strings"
)

func main() {

    scanner := bufio.NewScanner(os.Stdin)
    scanner.Scan()
    input := scanner.Text()

    parts := strings.Fields(input)

    var (
        totalSuara int
        suaraValid int
        suaraCalon = make(map[int]int)
    )

    for _, part := range parts {
        num, err := strconv.Atoi(part)
        if err != nil {
            continue
        }

        totalSuara++

        if num == 0 {
            break
        }

        if num >= 1 && num <= 20 {
            suaraValid++
            suaraCalon[num]++
        }
```

```
    }

    var calonTerpilih []int
    for calon := range suaraCalon {
        calonTerpilih = append(calonTerpilih, calon)
    }
    sort.Ints(calonTerpilih)

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

    for _, calon := range calonTerpilih {
        fmt.Printf("%d: %d\n", calon, suaraCalon[calon])
    }
}
```

OUTPUT:



DESKRIPSI:aplikasi penghitungan suara pemilihan

2.

```
// M.HANIF AL FAIZ
// 103112400042
package main

import (
    "bufio"
    "fmt"
    "os"
    "sort"
    "strconv"
    "strings"
)

type Calon struct {
    Nomor int
    Suara int
}

func main() {
    scanner := bufio.NewScanner(os.Stdin)
    scanner.Scan()
    input := scanner.Text()

    parts := strings.Fields(input)

    var (
        totalSuara int
        suaraValid int
        suaraCalon = make(map[int]int)
    )

    for _, part := range parts {
        num, err := strconv.Atoi(part)
        if err != nil {
            continue
        }

        totalSuara++

        if num == 0 {
            break
        }

        if num >= 1 && num <= 20 {
```

```
            suaraValid++
            suaraCalon[num]++
        }
    }

    var calons []Calon
    for nomor, suara := range suaraCalon {
        calons = append(calons, Calon{Nomor: nomor, Suara: suara})
    }

    sort.Slice(calons, func(i, j int) bool {
        if calons[i].Suara == calons[j].Suara {
            return calons[i].Nomor < calons[j].Nomor
        }
        return calons[i].Suara > calons[j].Suara
    })

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

    if len(calons) == 0 {
        fmt.Println("Tidak ada calon yang mendapatkan suara")
    } else if len(calons) == 1 {
        fmt.Printf("Ketua RT: %d\n", calons[0].Nomor)
        fmt.Println("Wakil ketua: Tidak ada")
    } else {
        fmt.Printf("Ketua RT: %d\n", calons[0].Nomor)
        fmt.Printf("Wakil ketua: %d\n", calons[1].Nomor)
    }
}
```

OUTPUT:

```
PS D:\103112400042_MODUL5> go run c:\Users\HP\Downloads\latihan2[1].go
7 19 3 2 78 3 1 -3 18 19 0
Suara masuk: 11
Suara sah: 8
Ketua RT: 3
Wakil ketua: 19
```

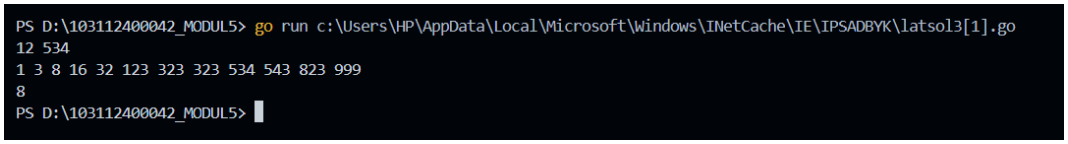
DESKRIPSI:menghitung hasil pemilihan Ketua dan Wakil Ketua RT berdasarkan suara yang masuk.

3.

```
//M.HANIF AL FAIZ
//103112400042
package main

import "fmt"

const NMAX = 1000000

var data [NMAX]int

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

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

    if pos == -1 {
        fmt.Println("TIDAK ADA")
    } else {
        fmt.Println(pos)
    }
}

func isiArray(n int) {
    /* I.S. terdefinisi integer n, dan sejumlah n data sudah siap pada piranti
masukan.
       F.S. Array data berisi n (<=NMAX) bilangan */
    for i := 0; i < n; i++ {
        fmt.Scan(&data[i])
    }
}

func posisi(n, k int) int {
    /* mengembalikan posisi k dalam array data dengan n elemen. Posisi dimulai
dari
       posisi 0. Jika tidak ada kembalikan -1 */
    left := 0
    right := n - 1

    for left <= right {
        mid := left + (right-left)/2

        if data[mid] == k {
```

```
            return mid
        } else if data[mid] < k {
            left = mid + 1
        } else {
            right = mid - 1
        }
    }

    return -1
}
```

OUTPUT:



DESKRIPSI:
melakukan pencarian bilangan dalam array yang sudah terurut menggunakan algoritma binary search

IV. KESIMPULAN

Dokumen ini menjelaskan implementasi dua metode pencarian, yaitu Sequential Search dan Binary Search, dalam bahasa Go, yang menekankan efisiensi dan kesederhanaan. Sequential Search digunakan untuk data tidak terurut, sementara Binary Search memerlukan data terurut namun lebih cepat dengan kompleksitas O(log n). Contoh aplikasinya meliputi pencarian nilai dalam array, data mahasiswa berdasarkan nama atau NIM, serta penghitungan suara pemilihan dan pemilihan Ketua RT. Binary Search terbukti efektif untuk data terurut, seperti pada pencarian bilangan dalam array. Dengan fitur Go seperti pengelolaan memori otomatis dan sintaksis sederhana, dokumen ini menunjukkan bagaimana algoritma pencarian dapat diimplementasikan secara optimal untuk menyelesaikan masalah praktis.

V. REFERENSI

Programmer Zaman Now. "ALGORITMA & STRUKTUR DATA - SEMESTER 2." YouTube, 2023, https://youtu.be/IO_vkyJnMas?si=4jSJTZ5Zd2Fi5Vd2.