LAPORAN PRAKTIKUM

ALGORITMA DAN PEMROGRAMAN 2

MODUL 8

“Sequential Binary”

DISUSUN OLEH:

SHEILA STEPHANIE ANINDYA

103112400086

S1 IF-12-01

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

I. DASAR TEORI

A. Sequential Search

Sequential search adalah teknik pencarian data yang dilakukan dengan cara membandingkan setiap elemen data satu per satu, mulai dari elemen pertama hingga elemen yang dicari ditemukan. Proses ini terus berlanjut hingga data ditemukan atau seluruh elemen telah diperiksa.

Algoritma ini termasuk salah satu algoritma pencarian yang paling sederhana dan sering digunakan dalam situasi di mana data tidak memiliki struktur tertentu.

Algoritma Sequential Search bekerja dengan melakukan perbandingan satu persatu secara beruntun dalam kumpulan data dengan data yang dicari sampai data tersebut ditemukan atau tidak ditemukan.

Proses pencarian ini hanya melakukan pengulangan data dari 1 sampai dengan jumlah data (N). Setiap pengulangan, dilakukan perbandingan data ke-i dengan data yang sedang dicari. Jika data sama dengan yang dicari, berarti data telah berhasil ditemukan. Sebaliknya, jika sampai akhir pengulangan tidak ada data yang sama dengan yang dicari, berarti data tidak ditemukan.

B. Binary Search

Binary Search merupakan sebuah teknik pencarian data dengancara berulang kali membagi separuh dari jumlah data yang dicari sampai sehingga memperkecil lokasi pencarian menjadi satu data. Dengan teknik ini kita akanmembuang setengah dari jumlah data. Apabila ditemukan kecocokan data maka program akan mengembalikan output, jika tidak pencarian akan terus berlanjut hingga akhir dari pembagian jumlah data tersebut. Algotihma ini biasanya banyak digunakan untuk mencari di program dengan jumlah data yang banyak, dimana kompleksitas dari algorithma ini adalah O(log n) di mana n adalah jumlah item. Pada saat menggunakan binary search, data yang berada di dalam array harus diurutkan terlebih dahulu.

Misalkan kita memiliki int arr[] = {70, 60, 30, 50, 40,20}, data para int arr harus diurutkan terlebih dahulu menggunakan teknik sorting seperti bubble sort. Sehingga array kita akan menjadi int arr[] = {20,30,40,50,60,70}.

II. GUIDED

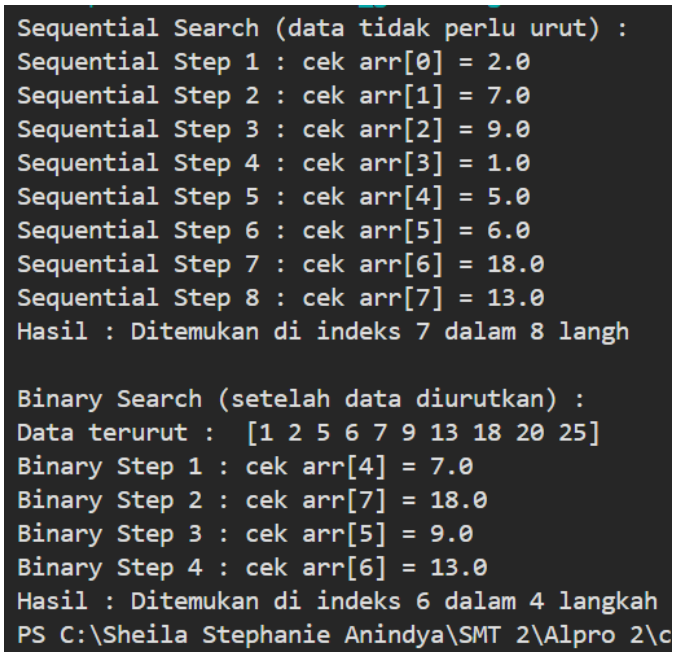
1. Source Code:

```
package main

import (
    "fmt"
    "sort"
)

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

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() {
    // 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 langh\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:



Penjelasan :

Program ini membandingkan efisiensi sequential search dan binary search dalam mencari nilai target pada sebuah array. Program menjalankan sequential search dengan mengecek elemen satu per satu hingga target ditemukan (mengembalikan indeks) atau tidak (-1). Selanjutnya, binary search mengurutkan array terlebih dahulu,

lalu membagi array menjadi dua bagian secara berulang untuk menemukan target lebih cepat.

2. Source Code:

```
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
```

Penjelasan :

Program mendata mahasiswa di array dan menerapkan sequential search untuk mencari mahasiswa berdasarkan nama, dan binary search untuk mencari berdasarkan NIM setelah data diurutkan. Program mengisi data 10 mahasiswa secara manual, kemudian mencari nama "Fajar" dengan sequential search dan NIM "2206" dengan binary search, sambil mencetak hasil indeks ditemukan dan jumlah iterasi pencarian untuk masing-masing metode.

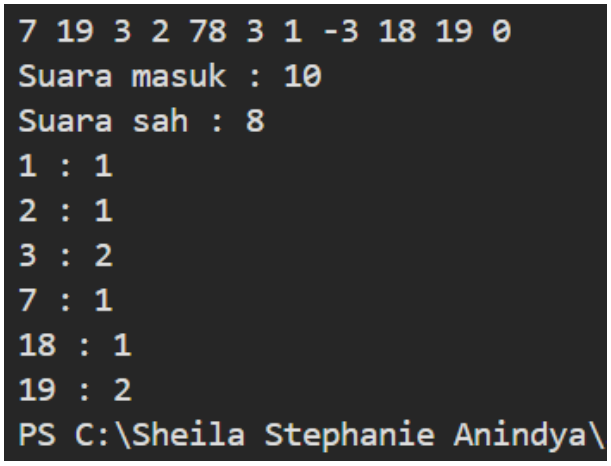
III. UNGUIDED

1. Source Code:

```
package main

import "fmt"

func main() {
    var (
        dataSuara []int
        a         int
        total     int
    )

    for {
        fmt.Scan(&a)
        if a == 0 {
            break
        }
        dataSuara = append(dataSuara, a)
        total++
    }

    hitungHasilPemilihan(dataSuara, total)
}

func hitungHasilPemilihan(suara []int, total int) {
    var (
        jumlahSuaraValid int
        tercatat         [21]bool
    )
```

```
    //hitung suara sah
    for i := 0; i < total; i++ {
        s := suara[i]
        if s >= 1 && s <= 20 {
            if !tercatat[s] {
                tercatat[s] = true
            }
            jumlahSuaraValid++
        }
    }

    fmt.Printf("Suara masuk : %d\n", total)
    fmt.Printf("Suara sah : %d\n", jumlahSuaraValid)

    for calon := 1; calon <= 20; calon++ {
        if tercatat[calon] {
            jumlah := hitungSuaraCalon(suara, total, calon)
            fmt.Printf("%d : %d\n", calon, jumlah)
        }
    }
}
func hitungSuaraCalon(data []int, panjang int, calon int) int {
    jumlah := 0
    for i := 0; i < panjang; i++ {
        if data[i] == calon {
            jumlah++
        }
    }
    return jumlah
}
```

Output:



Penjelasan :

Program scan input angka yang mewakili suara untuk calon nomor1 sampe 20, dan berhenti saat pengguna memasukkan angka 0. Data suara disimpan lalu dihitung jumlah suara total dan suara yang sah (yang bernilai 1 sampai 20). Setelah itu, program menampilkan total suara masuk, jumlah suara sah, serta jumlah suara yang diterima masing-masing calon yang mendapatkan suara.

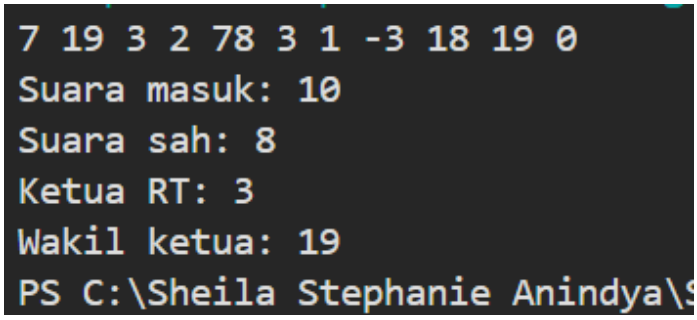
2. Source Code:

```
package main

import "fmt"

func main() {
    var (
        dataSuara []int
        a         int
        total     int
    )
```

```
    for {
        fmt.Scan(&a)
        if a == 0 {
            break
        }
        dataSuara = append(dataSuara, a)
        total++
    }

    hitungHasilPemilihan(dataSuara, total)
}

func hitungHasilPemilihan(suara []int, total int) {
    var (
        jumlahSuaraValid int
        suaraPerCalon    [21]int
    )

    //hitung suara sah
    for i := 0; i < total; i++ {
        s := suara[i]
        if s >= 1 && s <= 20 {
            suaraPerCalon[s]++
            jumlahSuaraValid++
        }
    }

    //calon dengan suara terbanyak (ketua RT)
    var ketuaRT, wakilKetua int
```

```
    var suaraKetua, suaraWakil int

    for calon := 1; calon <= 20; calon++ {
        if suaraPerCalon[calon] > suaraKetua {
            wakilKetua = ketuaRT
            suaraWakil = suaraKetua
            ketuaRT = calon
            suaraKetua = suaraPerCalon[calon]
        } else if suaraPerCalon[calon] > suaraWakil {
            wakilKetua = calon
            suaraWakil = suaraPerCalon[calon]
        }
    }

    fmt.Printf("Suara masuk: %d\n", total)
    fmt.Printf("Suara sah: %d\n", jumlahSuaraValid)
    fmt.Printf("Ketua RT: %d\n", ketuaRT)
    fmt.Printf("Wakil ketua: %d\n", wakilKetua)
}
```

Output:



Penjelasan :

Program menghitung hasil pemilihan Ketua dan Wakil Ketua RT berdasarkan input suara yang dimasukkan secara berulang sampai menginput angka 0. Setiap suara harus berupa angka antara 1 sampai 20, yang mewakili nomor calon; angka selain 1-20 dianggap tidak sah. Program menghitung total suara masuk, suara sah, serta jumlah suara yang diperoleh setiap calon, lalu menentukan dua calon dengan suara terbanyak sebagai Ketua dan Wakil Ketua RT.

3. Source Code:

```
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
        if data[tengah] == k {
            return tengah
        } else if data[tengah] < k {
            kiri = tengah + 1
        } else {
            kanan = tengah - 1
        }
    }
    return -1
}
```

Output:

```
12 534
1 3 8 16 32 123 323 323 534 543 823 999
8
PS C:\Sheila Stephanie Anindya\SMT 2\Alpro 2\
```

```
12 535
1 3 8 16 32 123 323 323 534 543 823 999
TIDAK ADA
PS C:\Sheila Stephanie Anindya\SMT 2\Alpro 2\
```

Penjelasan :

Program scan array berisi n angka yang diasumsikan sudah terurut, lalu mencari posisi (indeks) dari sebuah angka k menggunakan binary search. Data dimasukkan ke dalam array global data, dan fungsi posisi akan mengembalikan indeks di mana k ditemukan, atau -1 jika tidak ada. Jika ditemukan, program mencetak indeks tersebut; jika tidak, mencetak "TIDAK ADA".

IV. RINGKASAN

Sequential search adalah metode pencarian sederhana yang memeriksa setiap elemen satu per satu hingga target ditemukan atau seluruh data habis diperiksa. Cocok untuk data tidak terurut, tetapi kurang efisien untuk data besar karena kompleksitas waktu. Binary search lebih efisien (O(log n)) dengan syarat data harus terurut. Algoritma ini membagi data menjadi dua bagian secara berulang, membandingkan nilai tengah dengan target, dan mengeliminasi setengah data yang tidak relevan. Contohnya, array [70, 60, 30, 50, 40, 20] harus diurutkan menjadi [20, 30, 40, 50, 60, 70] sebelum pencarian. Binary search ideal untuk data besar, tetapi memerlukan pengurutan awal. Keduanya mengembalikan indeks jika target ditemukan atau -1 jika tidak, dengan binary search menang dalam kecepatan untuk dataset besar.

V. DAFTAR PUSTAKA

1. BINUS University. (2019, Desember 26). *Binary Search*.

https://socs.binus.ac.id/2019/12/26/binary-search/

2. FIKTI UMSU. (n.d.). *Algoritma Sequential Search: Pengertian, Fungsi, dan Cara Kerjanya*.

https://fikti.umsu.ac.id/algoritma-sequential-search-pengertian-fungsi-dan-cara-kerjanya/