LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2
MODUL 8
“PENCARIAN NILAI ACAK PADA HIMPUNAN DATA”

DISUSUN OLEH:

Muhammad Shabrian Fadly

103112400087

S1 IF-12-01

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024/2025

I. DASAR TEORI

1. Sequential Search

   Pencarian secara sekuensial ini adalah pencarian yang dilakukan dari data pertama, kedua hingga terakhir secara satu persatu dan berurutan. Ciri khas dari pencarian ini adalah proses pencarian akan berhenti ketika data yang dicari ditemukan, walaupun masih terdapat data yang belum dicek nilainya. Algoritma ini dikenal dengan nama *Sequential Search*, karena prosesnya melakukan pengecekan setiap elemen array secara satu persatu dan sekuensial dari data pertama hingga ditemukan atau data terakhir.

2. Binary Search

   Ide algoritma adalah: (dengan asumsi data terurut dari kecil membesar (ascending), dan data dengan indeks kecil ada di "kiri" dan indeks besar ada di "kanan")
   1) Ambil salah satu data dalam rentang data yang ada, algoritma di bawah menggunakan rentang
   dari kiri $\boldsymbol{kr}$ s.d. kanan $\boldsymbol{kn}$. Untuk kemudahan dan alasan lainnya, biasanya yang diambil adalah paling tengah dalam rentang tersebut.
   2) Jika data terambil tersebut terlalu kecil, maka ubah/geser rentang data ke sebelah kanan posisi data tersebut. Ini karena jika data terambil terlalu kecil, maka semua data sebalah kirinya juga akan terlalu kecil dari yang ingin dicari.
   3) Begitu juga sebaliknya jika data terambil terlalu besar.

   Algoritma ini dikenal dengan nama Binary Search.

3. Pencarian pada Array Bertipe Data Struct

   Algoritma pencarian pada array bertipe data struct tidak berbeda jauh dengan tipe data dasar, kita hanya perlu menambahkan field kategori dari pencarian yang akan dilakukan. Khusus untuk algoritma binary search, maka keterurutan array harus sama dengan field kategori dari pencariannya. Misalnya apabila pencarian berdasarkan nim mahasiswa, maka algoritma binary search hanya akan bisa digunakan apabila array terurut berdasarkan nim. Algoritma binary search tidak akan berjalan apabila array terurut berdasarkan nim mahasiswa, tetapi pencarian yang dilakukan berdasarkan filed nama atau kategori yang lain selain nim. Solusinya adalah array harus diurutkan berdasarkan nim terlebih dahulu, atau gunakan algoritma sequential search.

II. GUIDED

Source Code Guided 1:

```
//103112400087_Muhammad Shabrian
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 :

```
PS D:\Coding manja> go run "d:\Coding manja\modul8\guided1.go"
Squential Search (data tidak perlu urut)
Squential Step 1: cek arr[0] = 2.0
Squential Step 2: cek arr[1] = 7.0
Squential Step 3: cek arr[2] = 9.0
Squential Step 4: cek arr[3] = 1.0
Squential Step 5: cek arr[4] = 5.0
Squential Step 6: cek arr[5] = 6.0
Squential Step 7: cek arr[6] = 18.0
Squential 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: Program ini menunjukkan perbandingan antara dua cara mencari angka tertentu di dalam sebuah array. Cara pertama adalah sequential search, yang mencari satu per satu dari awal sampai ketemu. Cara kedua adalah binary search, yang lebih cepat tapi hanya bisa dilakukan kalau datanya sudah diurutkan. Program akan menampilkan langkah-langkah pencarian dan memberitahu kita apakah datanya ditemukan dan di posisi mana.

Source Code Guided 2:

```
//103112400087_Muhammad Shabrian
package main


import (
    "fmt"
    "sort"
)
type mahasiswa struct {
    nama, nim, kelas, jurusan string
    ipk float64
}
type arrMhs [2023]mahasiswa
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
}
```

```
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
    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},
    }
    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)
    }

    sort.Slice(data[:n], func(i, j int) bool {
        return data[i].nim < data[j].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:

```
PS D:\Coding manja> go run "d:\Coding manja\modul8\guided2.go"
Sequential Search - Cari nama 'Fajar'
Ditemukan di indeks: 5, Iterasi: 6

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

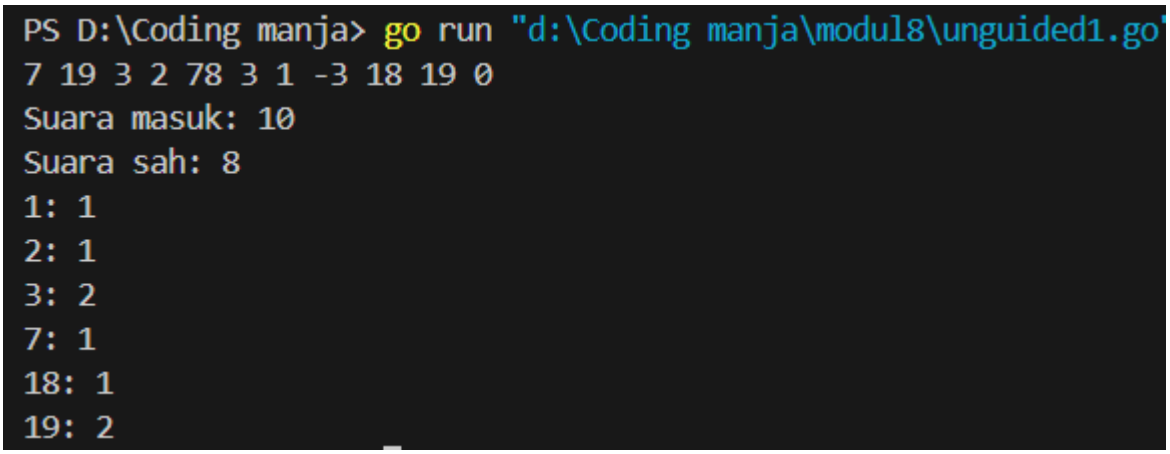
Deskripsi Program: Program ini digunakan untuk mencari data mahasiswa berdasarkan nama menggunakan sequential search, dan berdasarkan NIM menggunakan binary search. Pertama, data mahasiswa dibuat secara manual. Lalu, program mencari data mahasiswa yang dimaksud dan menampilkan di indeks berapa mahasiswa tersebut ditemukan serta berapa kali proses pencarian dilakukan. Ini membantu kita melihat efektivitas dua metode pencarian dalam data struct.

III. UNGUIDED

Source Code Unguided 1:

```
//103112400087_Muhammad Shabrian
package main
import "fmt"
func main() {
    var suara, totalMasuk, totalSah int
    var count [21]int
    for {
        fmt.Scan(&suara)
        if suara == 0 {
            break
        }
        totalMasuk++
        if suara >= 1 && suara <= 20 {
            count[suara]++
            totalSah++
        }
    }
    fmt.Println("Suara masuk:", totalMasuk)
    fmt.Println("Suara sah:", totalSah)
    for i := 1; i <= 20; i++ {
        if count[i] > 0 {
            fmt.Printf("%d: %d\n", i, count[i])
        }
    }
}
```

Output:



Deskripsi Program: Program ini dipakai untuk menghitung jumlah suara yang masuk dalam pemilihan ketua RT. Input berupa angka-angka pilihan warga, tapi hanya angka 1 sampai 20 yang dianggap sah. Program akan mencetak berapa jumlah suara yang masuk, berapa yang sah, dan berapa suara yang diterima oleh tiap calon.

Source Code Unguided 2 :

```
//103112400087_Muhammad Shabrian
package main
import "fmt"
func main() {
    var suara int
    var totalMasuk, totalSah int
    var count [21]int

    for {
        fmt.Scan(&suara)
        if suara == 0 {
            break
        }
        totalMasuk++
        if suara >= 1 && suara <= 20 {
            count[suara]++
            totalSah++
        }
    }
    fmt.Println("Suara masuk:", totalMasuk)
    fmt.Println("Suara sah:", totalSah)

    var ketua, wakil int
    for i := 1; i <= 20; i++ {
        if count[i] > count[ketua] || (count[i] == count[ketua] && i < ketua) {
```

```
            wakil = ketua

            ketua = i

        } else if count[i] > count[wakil] || (count[i] == count[wakil] && i < wakil &&
i != ketua) {

            wakil = i

        }

    }

    fmt.Println("Ketua RT:", ketua)

    fmt.Println("Wakil ketua:", wakil)

}
```

Output:

```
PS D:\Coding manja> go run "d:\Coding manja\modul8\unguided2.go"
7 19 3 2 78 3 1 -3 18 19 0
Suara masuk: 10
Suara sah: 8
Ketua RT: 3
Wakil ketua: 19
```

Deskripsi Program: Program ini melanjutkan dari program sebelumnya, tapi kali ini tujuannya adalah mencari siapa yang menjadi ketua RT dan wakil ketua RT. Ketua adalah calon yang mendapat suara terbanyak, dan wakilnya adalah yang mendapat suara terbanyak kedua. Kalau ada yang sama jumlahnya, maka calon dengan nomor lebih kecil akan dipilih. Program akan mencetak siapa ketuanya dan siapa wakilnya.

Source Code unguided 3:

```
//103112400087_Muhammad Shabrian
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) {
    for i := 0; i < n; i++ {
        fmt.Scan(&data[i])
    }
}

func posisi(n, k int) int {
```

```
    low := 0
    high := n - 1

    for low <= high {
        mid := (low + high) / 2
        if data[mid] == k {
            return mid
        } else if data[mid] < k {
            low = mid + 1
        } else {
            high = mid - 1
        }
    }
    return -1
}
```

Output:

```
PS D:\Coding manja> go run "d:\Coding manja\modul8\unguided3.go"
12 534
1 3 8 16 32 123 323 323 534 543 823 999
8
PS D:\Coding manja> go run "d:\Coding manja\modul8\unguided3.go"
12 535
1 3 8 16 32 123 323 323 534 543 823 999
TIDAK ADA
```

Deskripsi Program: Program ini menerima sejumlah data angka yang sudah dalam keadaan terurut dari kecil ke besar, lalu mencari apakah angka tertentu ada di dalamnya atau tidak. Kalau ada, program akan mencetak posisi angkanya. Kalau tidak ada, maka akan muncul tulisan "TIDAK ADA". Program menggunakan metode binary search karena lebih cepat untuk data besar yang sudah urut.

IV. KESIMPULAN

Dari praktikum tersebut, dapat disimpulkan bahwa algoritma pencarian memegang peranan yang sangat penting dalam mengolah data, yaitu menemukan data yang dicari dalam suatu kumpulan data. Algoritma sequential search yang bekerja dengan mengechek elemen satu per satu, cocok untuk data yang belum terurut, sedangkan binary search lebih cepat namun mensyaratkan data dalam kondisi terurut. Selain tersebut pula, pencarian pada array bertipe struct menunjukkan bahwa kita dapat melakukan pencarian berdasarkan field tertentu, akan tetapi algoritma binary search hanya dapat digunakan bila data sudah terurut berdasarkan field pencarian tersebut. Aplikasi dari masing-masing algoritma pencarian tersebut juga telah terbukti efektif, baik ketika digunakan untuk studi kasus sederhana seperti pemilihan ketua RT ataupun pencarian bilangan pada array besar. Oleh karena itu, pemahaman dan penggunaan algoritma pencarian menjadi hal yang penting dalam penyelesaian masalah komputasi sehari-hari.

V. REFERENSI

Modul praktikum algoritma dan pemrograman 2: Modul 11. Pencarian Nilai Acak Pada Himpunan Data