LAPORAN PRAKTIKUM

ALGORITMA DAN PEMROGRAMAN 2

MODUL 8

“Sequential Binary”

DISUSUN OLEH:

RAJA MUHAMMAD LUFHTI

103112400027

S1 IF-12-01

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

I. DASAR TEORI

Pada praktikum ini, pembahasan difokuskan pada penerapan rekursi dalam pemrograman menggunakan bahasa C. Rekursi adalah suatu teknik pemrograman di mana suatu fungsi memanggil dirinya sendiri untuk menyelesaikan masalah yang lebih kecil dari masalah utama. Konsep ini sangat berguna dalam pemecahan masalah yang memiliki struktur berulang atau bersifat hierarkis, seperti perhitungan faktorial, deret Fibonacci, dan pencarian dalam struktur data rekursif (misalnya pohon atau grafik).

Fungsi rekursif terdiri dari dua bagian penting:

1. Basis kasus (base case): kondisi yang menghentikan pemanggilan rekursif.
2. Panggilan rekursif (recursive call): bagian fungsi yang memanggil dirinya sendiri untuk menyelesaikan sub-masalah.

Keunggulan rekursi adalah menyederhanakan penulisan kode untuk masalah tertentu, namun penggunaan yang tidak efisien atau basis kasus yang tidak jelas dapat menyebabkan program mengalami kegagalan (*stack overflow*). Oleh karena itu, pemahaman konsep rekursi sangat penting agar program yang dibuat dapat berjalan dengan optimal dan tidak mengalami kesalahan.

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:

```
PS D:\ALPRO 2\103112400027_MODUL> go run "d:\ALPRO 2\103112400027_MODUL\g1\1.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 pada index 7 dalam 8 langkah


Binary Search (setelah data di urutkan):
Data urut: [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 pada index 6 dalam 4 langkah
PS D:\ALPRO 2\103112400027_MODUL>
```

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:

```
PS D:\ALPRO 2\103112400027_MODUL> go run "d:\ALPRO 2\103112400027_MODUL\g2\2.go"
Sequential Search - Cari nama 'Fajar'
Ditemukan di indeks: 5, Iterasi: 6

Binary Search - Cari NIM '2206'
Ditemukan di indeks: 5, Iterasi: 3
PS D:\ALPRO 2\103112400027_MODUL>
```

Penjelasan :

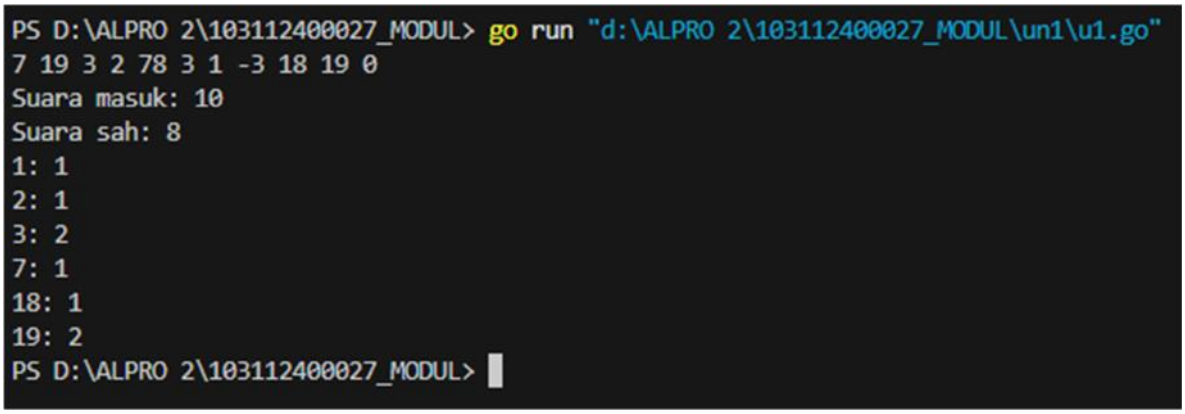
Program mendata mahasiswa di array dan menerapkan sequential search untuk mencari mahasiswa berdasarkan nama, dan binary search untuk mencari berdasarkan NIM setelah data diurutkan. Program mengisi data 10 mahasiswa secara manual, kemudian mencari nama "Fajar" dengan sequential search dan NIM "2206"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

III. UNGUIDED

1. Source Code:

```
//RAJA_103112400027

package main

import (
    "fmt"
)

const N = 1000
const MAX_CALON = 20

func main() {
    var suara [N]int
    var jumlahSuara [MAX_CALON + 1]int
    var totalMasuk, totalSah int
    var input int

    for {
        fmt.Scan(&input)
        if input == 0 {
            break
        }
        suara[totalMasuk] = input
        totalMasuk++
    }

    for i := 0; i < totalMasuk; i++ {
        if suara[i] >= 1 && suara[i] <= MAX_CALON {
            jumlahSuara[suara[i]]++
            totalSah++
        }
    }

    fmt.Println("Suara masuk:", totalMasuk)
    fmt.Println("Suara sah:", totalSah)
    for i := 1; i <= MAX_CALON; i++ {
        if jumlahSuara[i] > 0 {
            fmt.Printf("%d: %d\n", i, jumlahSuara[i])
        }
    }
}
```

Output:



Penjelasan :

Program membaca bilangan satu per satu hingga 0 ditemukan.Nilai yang valid (1–20) dianggap sebagai suara sah dan disimpan jumlahnya. Program mencetak jumlah total suara masuk, jumlah suara sah, dan suara yang didapatkan tiap calon.

2. Source Code:

```
//RAJA_103112400027
package main

import (
    "fmt"
)

const MAX_CALON = 20
const N = 1000
```

```
func main() {
    var suara [N]int
    var jumlahSuara [MAX_CALON + 1]int
    var totalMasuk, totalSah int
    var input int

    for {
        fmt.Scan(&input)
        if input == 0 {
            break
        }
        suara[totalMasuk] = input
        totalMasuk++
    }

    for i := 0; i < totalMasuk; i++ {
        if suara[i] >= 1 && suara[i] <= MAX_CALON {
            jumlahSuara[suara[i]]++
            totalSah++
        }
    }

    ketua, wakil := -1, -1
    max1, max2 := 0, 0

    for i := 1; i <= MAX_CALON; i++ {
        if jumlahSuara[i] > max1 {
            max2 = max1
            wakil = ketua
            max1 = jumlahSuara[i]
            ketua = i
        } else if jumlahSuara[i] == max1 && ketua != -1 && i < ketua {
            wakil = ketua
            ketua = i
        } else if jumlahSuara[i] > max2 && jumlahSuara[i] < max1 {
            max2 = jumlahSuara[i]
            wakil = i
        } else if jumlahSuara[i] == max2 && i < wakil {
            wakil = i
        }
    }

    fmt.Println("Suara masuk:", totalMasuk)
    fmt.Println("Suara sah:", totalSah)
    fmt.Println("Ketua RT:", ketua)
    fmt.Println("Wakil ketua:", wakil)
}
```

Output:


PS D:\ALPRO 2\103112400027_MODUL> go run "d:\ALPRO 2\103112400027_MODUL\un2\u2.go"
7 19 3 2 78 3 1 -3 18 19 0
Suara masuk: 10
Suara sah: 8
Ketua RT: 3
Wakil ketua: 1
PS D:\ALPRO 2\103112400027_MODUL>

Penjelasan :

Program ini dibuat untuk menghitung hasil pemilihan ketua RT dengan membaca serangkaian suara yang diberikan dalam bentuk angka dan memvalidasi suara sah (antara 1–20), kemudian menentukan siapa yang menjadi ketua dan wakil ketua berdasarkan jumlah suara terbanyak. Jika terdapat lebih dari satu calon dengan jumlah suara yang sama, maka dipilih calon dengan nomor terkecil sebagai ketua dan wakil ketua. Proses ini dilakukan dengan menggunakan array untuk menyimpan jumlah suara tiap calon, serta logika pemilihan berjenjang sesuai materi pencarian dan validasi data pada Modul 11 Algoritma dan Pemrograman 2.

3. Source Code:

```
//RAJA_103112400027
package main

import "fmt"

const NMAX = 1000000
var data [NMAX]int

func main() {
   var n, k int

   fmt.Scan(&n, &k)

   isiArray(n)

   index := posisi(n, k)

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

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

```
func posisi(n, k int) int {
    kiri := 0
    kanan := n - 1
    var tengah int

    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:

```
PS D:\ALPRO 2\103112400027_MODUL> go run "d:\ALPRO 2\103112400027_MODUL\un3\u3.go"
12 534
1 3 8 16 32 123 323 323 534 543 823 999
8
PS D:\ALPRO 2\103112400027_MODUL> go run "d:\ALPRO 2\103112400027_MODUL\un3\u3.go"
12 535
1 3 8 16 32 123 323 323 534 543 823 999
TIDAK ADA
PS D:\ALPRO 2\103112400027_MODUL>
```

Penjelasan :

Program ini digunakan untuk mencari posisi suatu bilangan kk dalam sekumpulan data integer yang telah terurut membesar dengan menggunakan algoritma **binary search** sesuai materi pada Modul 11. Program membaca jumlah data nn dan nilai yang dicari kk, lalu mengisi array dengan nn bilangan dari input. Fungsi posisi melakukan pencarian dengan membandingkan kk terhadap elemen tengah array dan mempersempit rentang pencarian hingga data ditemukan atau rentang habis. Jika ditemukan, program mencetak indeksnya; jika tidak, mencetak "TIDAK ADA". Pendekatan ini efisien karena kompleksitas waktunya logaritmik, sesuai dengan struktur data yang sudah terurut.

IV. KESIMPULAN

Dari praktikum yang dilakukan, dapat disimpulkan bahwa:

- Rekursi merupakan metode pemrograman yang efektif untuk menyelesaikan masalah yang bersifat berulang atau dapat dipecah menjadi sub-masalah serupa.
- Dalam menggunakan rekursi, penting untuk menentukan basis kasus yang tepat agar fungsi tidak memanggil dirinya secara tak terbatas.
- Pemahaman tentang alur eksekusi fungsi rekursif sangat penting agar mahasiswa dapat merancang dan menerapkan solusi yang tepat menggunakan teknik ini.
- Praktikum ini membantu mahasiswa dalam memahami konsep rekursi secara praktis dan bagaimana mengimplementasikannya pada berbagai jenis permasalahan.

V. DAFTAR PUSTAKA

MODUL ALGORITMA PEMROGRAMAN