

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

MODUL 12

PENCARIAN DATA/SEARCHING



Oleh: Dimas Fanny Hebrasianto Permadi

NAMA: Dimas Ramadhani

NIM: 103112400065

KELAS: IF-12-01

**S1 TEKNIK INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO**

2025

I. DASAR TEORI

1. Sequential Search

Pencarian secara sequential adalah pencarian yang dilakukan satu-persatu dari array pertama sampai akhir. Ciri khas dari cara ini adalah jika ditemukan data yang dicari maka akan berhenti walaupun masih ada data setelahnya.

Cara kerjanya:

- Misalkan ada suatu array dengan indeks dari 0 hingga N-1, lalu mencari nilai X pada array tersebut.
- Digunakan status pencarian untuk menandakan data yang dicari ditemukan atau tidak, misalkan variabel found dengan tipe boolean
- Pencarian dilakukan dari array indeks ke-0 sampai ke-N-1, setiap kali perbandingan dengan X, update nilai found.
- Perulangan harus dihentikan apabila status pencarian found bernilai true (data ditemukan) atau Array[N-1] telah dicek.

2. Binary Search

Pada pencarian Binary Search, variabel array harus sudah terurut dari kecil membesar (ascending) atau terurut besar mengecil (descending), dan data dengan indeks kecil ada di “kiri” dan indeks besar ada di “kanan”.

- Menentukan indeks awal dan akhir array
- Hitung indeks tengah $mid = \frac{awal+akhir}{2}$
- Jika elemen tengah sama dengan target, pencarian selesai
- Jika target lebih kecil dari elemen tengah, cari di bagian kiri (awal hingga mid-1)
- Jika lebih besar, cari di bagian kanan (mid+1 hingga akhir)

3. Pencarian pada array bertipe data struct

Algoritma pencarian pada array bertipe data struct tidak berbeda jauh dengan data dasar. Khusus untuk algoritma binary search, keterurutan array harus sama dengan field kategori dari pencariannya.

I. GUIDED

1. sequentialSearch1

- Source Code:

```
package main

import (
    "fmt"
    "sort"
)

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
}

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 perluurut):")
    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",
iterseq)
    }

    // Binary search perlu array diurutkan
    sort.Float64s(data)
    fmt.Println("BIinary Search (seteah 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)
    }
}

```

- **Screenshot Hasil Program:**

```
\Dimas\OneDrive\Collage\Semester 2\Algoritma dan
tma dan Pemrograman 2\Praktikum\Laparak\1031124000
Sequential Search (data tidak perluurut):
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 (seteah 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
```

- **Penjelasan:**

Program ini bertujuan untuk mencari sebuah nilai tertentu (target) dalam sebuah array angka bertipe float64, menggunakan dua metode pencarian berbeda yaitu sequential search (Pencarian Berurutan) dan Binary Search (Pencarian Biner). Program akan menampilkan langkah-langkah pencarian yang dilakukan, posisi indeks jika data ditemukan, dan jumlah langkah (iterasi) yang dilakukan oleh masing-masing metode.

Pada fungsi sequentialSearch akan menerima array arr dan nilai target untuk dicari. Lalu akan mengembalikan indeks tempat nilai ditemukan (atau -1 jika tidak ditemukan), dan jumlah langkah pencarian (iterations). Lalu fungsi meninisialisasikan variable penghitung langkah (iterations) dengan nilai 0. Lalu program akan melakukan perulangan for-range, i adalah indeks, val adalah nilai pada indeks ke-i. Setiap iterasi, penghitung langkah (iterations) bertambah. Lalu program menampilkan langkah saat ini, indeks, dan nilai yang dicek. Jika ditemukan, kembalikan indeks dan jumlah langkah. Jika selesai dicek dan tidak ditemukan, akan mengembalikan -1 dan jumlah langkah (iterations).

Pada fungsi `binarySearch` sama seperti `sequential`, tapi menggunakan logika pembagi dua. Inisialisasi variabel `iterations = 0` sebagai penghitung langkah, variabel `low = 0` dan `high = panjang array arr - 1` sebagai batas pencarian awal dan akhir. Lalu masuk ke perulangan dengan kondisi `low kurang dari sama dengan high`. Menambahkan nilai variabel `iterations` dan inisialisasi variabel $mid = \frac{low+high}{2}$. Lalu program akan menampilkan langkah saat ini. Lalu membandingkan nilai tengah dengan target. Jika sama, akan mengembalikan indeks. Jika `target < mid`, geser pencarian ke kiri. Jika lebih besar, geser ke kanan. Jika tidak ditemukan, return -1.

Pada fungsi utama, program membuat variabel `slice` berisi nilai acak, tidak terurut. Lalu variabel `data` sama dengan 13.0. Menampilkan info awal proses `sequential`. Fungsi ini akan memanggil fungsi `sequentialSearch` dan menyimpan hasilnya dalam `idxseq` sebagai indeks hasil dan `iterseq` sebagai jumlah langkah. Lalu fungsi ini akan mengecek apakah target ditemukan atau tidak, lalu tampilkan hasilnya. Lalu program mengurutkan data agar bisa digunakan oleh `binarySearch`. Menampilkan bahwa data sekarang sudah terurut. Lalu fungsi ini memanggil fungsi `binarySearch`. Menampilkan hasil pencarian `binarySearch`.

2. Nomor 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)
}

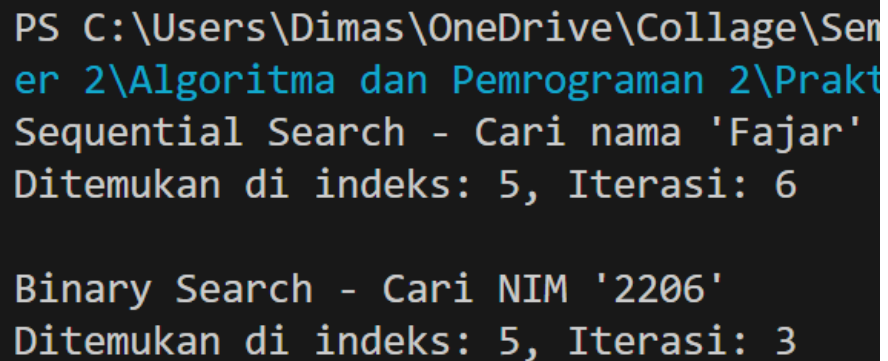
// Urutkan data berdasarkan NIM untuk binary search
sort.Slice(data[:n], func(i, j int) bool {
    return data[i].nim < data[j].nim
}))

// 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)
}
}

```

- **Screenshot Hasil Program:**



```
PS C:\Users\Dimas\OneDrive\Collage\Semester 2\Algoritma dan Pemrograman 2\Praktikum 1> Sequential Search - Cari nama 'Fajar'
Ditemukan di indeks: 5, Iterasi: 6

PS C:\Users\Dimas\OneDrive\Collage\Semester 2\Algoritma dan Pemrograman 2\Praktikum 1> Binary Search - Cari NIM '2206'
Ditemukan di indeks: 5, Iterasi: 3
```

- **Penjelasan:**

Program ini bertujuan untuk mencari data mahasiswa dalam sebuah array bertipe `arrMhs`, yang menyimpan informasi nama, NIM, kelas, jurusan, dan IPK. Program menggunakan dua metode pencarian data, yaitu sequential search berdasarkan nama mahasiswa dan binary search berdasarkan NIM mahasiswa. Program akan menampilkan langkah pencarian, posisi indeks jika data ditemukan, dan jumlah iterasi yang dilakukan oleh masing-masing metode pencarian.

Struktur data, mahasiswa berfungsi sebagai struct yang menyimpan informasi nama, NIM, kelas, jurusan, dan IPK. Lalu `arrMhs` sebagai array tetap bertipe mahasiswa sebanyak 2023 elemen.

Fungsi `seqSearch_3` (Pencarian Nama – Sequential Search), fungsi ini menerima array mahasiswa `T`, jumlah data `n`, dan nama `x` yang akan dicari. Fungsi ini akan mengembalikan indeks hasil pencarian dan jumlah langkah (iterasi). Inisialisasi variabel `found = -1` (jika tidak ditemukan), `j = 0` untuk iterasi, `iterasi = 0` untuk menghitung jumlah langkah. Fungsi melakukan perulangan `for` selama belum ditemukan dan indeks `j` masih dalam batas. Setiap iterasi akan ditambah dengan iterasi, jika `T[j].nama` sama dengan `x`, maka simpan posisi `j` ke `found`. Tambah `j`. Jika selesai dan tidak ditemukan, maka `found = -1` akan dikembalikan.

Fungsi `binarySearch_3` (Pencarian NIM – Binary Search). Fungsi ini menerima array `T`, jumlah data `n`, dan NIM `x` yang akan dicari. Fungsi mengembalikan indeks hasil pencarian dan jumlah iterasi. Inisialisasi variabel `found = -1` sebagai penanda tidak ditemukan, `kr = 0`, `kn = n-1` sebagai batas kiri dan kanan, `iterasi = 0`. Prosesnya

selama $kr \leq kn$ dan belum ditemukan, tambah iterasi, hitung $med = \frac{kr+kn}{2}$, jika x kurang dari $T[med].nim$ maka pencarian ke kiri ($kn = med - 1$), jika x lebih dari $T[med].nim$ maka pencarian ke kanan ($kr = med + 1$), jika sama maka $found = med$. Jika tidak ditemukan, mengembalikan nilai -1.

Fungsi main (pemanggilan dan eksekusi), inisialisasi array data dengan 10 data mahasiswa secara manual. Panggil fungsi `seqSearch_3` untuk mencari "Fajar". Hasil pencarian dicetak apakah ditemukan atau tidak, beserta jumlah iterasinya. Lakukan pengurutan data dengan `sort.Slice()` berdasarkan nim untuk keperluan binary search. Panggil fungsi `binarySearch_3` untuk mencari NIM "2206". Hasil pencarian ditampilkan di terminal.

II. UNGUIDED

1. Nomor 1

- Source Code:

```
package main

import (
    "fmt"
)
/*
Dimas Ramadhani
103112400065
*/
func pilkart() {
    var target int
    var suaraMasuk, suaraSah int = 0, 0
    calonKetua := make([]int, 20)
    jumlahVote := make([]int, 20)
    for i := 0; i < 20; i++ {
        calonKetua[i] = i + 1
    }
    for {
        _, err := fmt.Scan(&target)
        if err != nil {
            break
        }
        if target == 0 {
            break
        }
        suaraMasuk++
        for i, val := range calonKetua {
            if val == target {
                suaraSah++
                jumlahVote[i]++
                break
            }
        }
    }
    fmt.Println("Suara masuk:", suaraMasuk)
    fmt.Println("Suara sah:", suaraSah)
    for i := 0; i < len(calonKetua); i++ {
        if jumlahVote[i] > 0 {
            fmt.Printf("%d: %d\n", i+1, jumlahVote[i])
        }
    }
}
```

```
}  
}  
func main() {  
  pilkart()  
}
```

- **Screenshot Hasil Pembahasan:**

```
PS C:\Users\Dimas\OneDrive\Co
er 2\Algoritma dan Pemrograman
7 19 3 2 78 3 1 -3 18 19 0
Suara masuk: 10
Suara sah: 8
1: 1
2: 1
3: 2
7: 1
18: 1
19: 2
```

- **Penjelasan:**

Program ini bertujuan untuk mensimulasikan proses pemungutan suara dalam pemilihan ketua RT, di mana pemilih dapat memilih salah satu dari 20 calon ketua yang bernomor 1 hingga 20. Program membaca input pemilih, memvalidasi suara, menghitung total suara yang masuk dan yang sah, serta menampilkan hasil perolehan suara dari masing-masing calon ketua RT.

Fungsi pilkart melakukan deklarasi dan inisialisasi variabel target untuk menampung input suara (nomor calon ketua), suaraMasuk untuk menghitung total suara yang diberikan (termasuk tidak sah), suaraSah untuk menghitung total suara sah (bernilai antara 1-20), calonKetua untuk slice dengan panjang 20 yang diisi dengan nomor urut calon ketua dari 1 sampai 20, jumlahVote sebagai slice untuk menghitung jumlah suara masing-masing calon. Mengisi array calonKetua dengan angka 1 sampai 20 sebagai nomor urut calon. Program membaca input satu persatu hingga pengguna memasukkan angka 0 atau input tidak valid. suaraMasuk selalu bertambah untuk setiap input. Hanya jika target sesuai dengan salah satu nomor calon (1-20), maka suaraSah bertambah dan suara untuk calon tersebut

(jumlahVoter[i]) ditambah 1. Jika target di luar 1-20, suara dianggap tidak sah. Menampilkan jumlah total suara yang masuk. Menampilkan jumlah suara sah. Menampilkan hasil perolehan suara dari setiap calon yang mendapat suara.

Pada fungsi utama hanya memanggil fungsi pilkart saja.

2. Nomor 2

- Source Code:

```
package main

import (
    "fmt"
)

func pilkart() {
    var target int
    var suaraMasuk, suaraSah int = 0, 0
    calonKetua := make([]int, 20)
    jumlahVote := make([]int, 20)
    for i := 0; i < 20; i++ {
        calonKetua[i] = i + 1
    }
    for {
        _, err := fmt.Scan(&target)
        if err != nil {
            break
        }
        if target == 0 {
            break
        }
        suaraMasuk++
        for i, val := range calonKetua {
            if val == target {
                suaraSah++
                jumlahVote[i]++
                break
            }
        }
    }
    fmt.Println("Suara masuk:", suaraMasuk)
    fmt.Println("Suara sah:", suaraSah)
    /*
        Dimas Ramadhani
        103112400065
    */
    jumKetua1, jumKetua2 := 0, 0
}
```

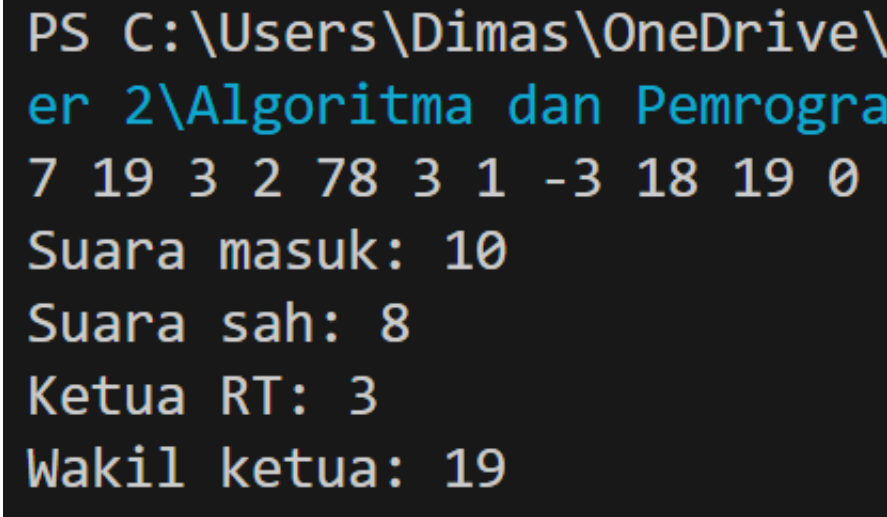


```

cekKetua1, cekKetua2 := -1, -1
for i, vote := range jumlahVote {
    if vote > jumKetua1 || (vote == jumKetua1 && cekKetua1
> i) {
        jumKetua2, cekKetua2 = jumKetua1, cekKetua1
        jumKetua1, cekKetua1 = vote, i
    } else if vote > jumKetua2 || (vote == jumKetua2 &&
cekKetua2 > i) {
        jumKetua2, cekKetua2 = vote, i
    }
}
if cekKetua1 != -1 {
    fmt.Println("Ketua RT:", cekKetua1+1)
}
if cekKetua2 != -1 {
    fmt.Println("Wakil ketua:", cekKetua2+1)
}
}
func main() {
    pilkart()
}

```

- **Screenshot Hasil Pembahasan:**



```

PS C:\Users\Dimas\OneDrive\er 2\Algoritma dan Pemrograman
7 19 3 2 78 3 1 -3 18 19 0
Suara masuk: 10
Suara sah: 8
Ketua RT: 3
Wakil ketua: 19

```

- **Penjelasan:**

Program ini bertujuan untuk melakukan simulasi pemilihan ketua RT dari 20 calon yang tersedia. Program akan menerima suara berupa nomor urut calon, menghitung total suara masuk dan suara sah, menampilkan hasil perolehan suara setiap calon yang mendapat suara, serta menentukan dua calon dengan suara terbanyak sebagai Ketua dan Wakil Ketua RT berdasarkan hasil pemungutan suara.

Fungsi pilkart melakukan deklarasi dan inisialisasi variabel target untuk menampung input suara (nomor calon ketua), suaraMasuk untuk menghitung total suara yang diberikan (termasuk tidak sah), suaraSah untuk menghitung total suara sah (bernilai antara 1-20), calonKetua untuk slice dengan panjang 20 yang diisi dengan nomor urut calon ketua dari 1 sampai 20, jumlahVote sebagai slice untuk menghitung jumlah suara masing-masing calon. Mengisi array calonKetua dengan angka 1 sampai 20 sebagai nomor urut calon. Program membaca input satu persatu hingga pengguna memasukkan angka 0 atau input tidak valid. suaraMasuk selalu bertambah untuk setiap input. Hanya jika target sesuai dengan salah satu nomor calon (1-20), maka suaraSah bertambah dan suara untuk calon tersebut (jumlahVoter[i]) ditambah 1. Jika target di luar 1-20, suara dianggap tidak sah. Menampilkan jumlah total suara yang masuk. Menampilkan jumlah suara sah. Mencari dua calon dengan suara tertinggi yaitu jumKetua1, cekketua1 untuk menyimpan suara dan indeks (nomor calon) ketua RT. jumKetua2 dan cekKetua2 untuk wakil ketua RT. Jika ada suara sama, calon dengan nomor lebih kecil diprioritaskan (menggunakan i sebagai indeks). Lalu menampilkan hasil pemilihan ketua dan wakil ketua RT berdasarkan suara terbanyak. Indeks cekKetua1 dan cekKetua2 ditambah 1 karena nomor calon mulai dari 1 (bukan 0).

Fungsi utama hanya memanggil pilkart untuk menjalankan seluruh proses.

3. Nomor 3

- Source Code:

```
package main

import "fmt"

const NMAX = 10000

var data [NMAX]int

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

    letak := posisi(n, k)
    if letak == -1 {
        fmt.Print("TIDAK ADA")
    } else {
        fmt.Print(letak)
    }
}

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

func posisi(a, b int) int {
    low := 0
    high := a - 1
    for low <= high {
        mid := (low + high) / 2
        if data[mid] == b {
            return mid
        } else if b < data[mid] {
            high = mid - 1
        } else {
            low = mid + 1
        }
    }
    return -1
}
```

```
}
```

```
/*
```

```
Dimas Ramadhani
```

```
103112400065
```

```
*/
```

- **Screenshoot Hasil Pembahasan:**

```
PS C:\Users\Dimas\OneDrive\Collage\Semester 2\Algoritma dan Pemrograman 2\Praktikum
12 534
1 3 8 16 32 123 323 323 534 543 823 999
8
PS C:\Users\Dimas\OneDrive\Collage\Semester 2\Algoritma dan Pemrograman 2\Praktikum
12 535
1 3 8 16 32 123 323 323 534 543 823 999
TIDAK ADA
```

- **Penjelasan:**

Program ini bertujuan untuk melakukan pencarian posisi elemen (nilai tertentu) dalam sebuah array menggunakan algoritma Binary Search. Data yang dicari diinput oleh pengguna bersama dengan data array-nya. Jika data ditemukan, program akan mencetak indeks (posisi) dari data tersebut dalam array; jika tidak, akan mencetak "TIDAK ADA".

Deklarasikan konstanta dan variabel global, NMAX adalah konstanta yang menyatakan ukuran maksimum array. Data menunjukkan array global bertipe int yang dapat menyimpan hingga 10.000 elemen.

Fungsi utama, menerima input dari pengguna seperti n untuk banyaknya elemen dalam array, variabel k adalah nilai yang ingin dicari. Memanggil fungsi isiArray untuk mengisi array data sebanyak n elemen. Memanggil fungsi posisi untuk melakukan pencarian nilai k dalam array data, dan menyimpan hasil indeks dalam variabel letak. Jika letak bernilai -1, berarti data tidak ditemukan, maka ditampilkan "TIDAK ADA". Jika ditemukan, cetak indeksnya (dari 0)

Fungsi `isiArray(a int)` untuk melakukan input sebanyak `a` data dari pengguna dan menyimpan ke dalam array global `data`.

Fungsi `posisi(a, b int)int`, variabel `low` dan `high` adalah batas bawah dan atas pencarian. Melakukan perulangan selama `low` masih kurang dari sama dengan `high`. `Mid` adalah indeks tengah dari interval saat ini. Jika elemen tengah adalah `b`, maka `mid` dikembalikan sebagai posisi ditemukan. Jika nilai `b` lebih kecil dari elemen tengah, pencarian difokuskan ke bagian kiri ($high = mid - 1$). Jika lebih besar, ke bagian kanan ($low = mid + 1$). Jika tidak ditemukan setelah pencarian, kembalikan `-1`.

III. KESIMPULAN

Pada praktikum ini, telah dipelajari dan diterapkan algoritma pencarian data, yaitu Sequential Search dan Binary Search, dalam bahasa pemrograman Go. Kedua algoritma ini digunakan untuk menemukan suatu elemen dalam sekumpulan data (array) berdasarkan nilai tertentu.

Penerapan algoritma ini mencakup dua jenis data, yaitu:

- Sequential Search, yang bekerja dengan cara memeriksa setiap elemen secara berurutan dari awal hingga akhir hingga data yang dicari ditemukan atau proses pencarian selesai.
- Binary Search, yang hanya dapat diterapkan pada data yang sudah terurut, dengan prinsip membandingkan nilai tengah array, lalu mempersempit ruang pencarian ke kiri atau kanan berdasarkan hasil perbandingan.

Pemahaman terhadap algoritma pencarian ini sangat penting dalam pengolahan data, karena memungkinkan proses identifikasi posisi elemen tertentu secara efisien. Praktikum ini juga memperkuat pemahaman mahasiswa mengenai perbedaan efisiensi antara algoritma pencarian sederhana dan pencarian yang lebih optimal dalam struktur data terurut.

IV. REFERENSI

Prayogo, N. A. (2024). *Dasar Pemrograman Golang* (Versi 4.0.20240830). Retrieved from <https://github.com/novalagung/dasarpemrogramangolang>

Selly Meliana, S.Kom., M.Kom.(2024) *Modul Praktikum ALGORITMA DAN PEMROGRAMAN 2*.