

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



Oleh:

NAMA: MOHAMMAD REYHAN ARETHA FATIN

NIM: 103112400078

KELAS: IF-12-01

S1 TEKNIK INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

I. DASAR TEORI

1. Sequential Search (Pencarian Sekuensial)

Sequential Search adalah salah satu metode pencarian yang paling sederhana, di mana elemen-elemen dalam array atau daftar diperiksa satu per satu secara berurutan hingga data yang dicari ditemukan atau seluruh data telah diperiksa. Proses pencarian dimulai dari elemen pertama dalam array dan berlanjut ke elemen berikutnya hingga data yang dicari ditemukan atau hingga akhir array tercapai. Kelebihan dari algoritma ini adalah dapat diterapkan pada array yang tidak terurut, sehingga memudahkan pencarian pada data yang tidak memiliki urutan tertentu. Namun, algoritma ini memiliki kelemahan dalam hal efisiensi, karena kompleksitas waktunya adalah $O(n)$, yang berarti waktu pencarian akan bertambah seiring dengan bertambahnya jumlah elemen dalam array. Oleh karena itu, meskipun mudah diterapkan, Sequential Search kurang efisien pada dataset yang besar.

2. Binary Search (Pencarian Biner)

Binary Search adalah algoritma pencarian yang lebih efisien dibandingkan dengan Sequential Search, namun hanya dapat digunakan pada array yang telah diurutkan. Algoritma ini bekerja dengan membagi array yang terurut menjadi dua bagian dan memilih bagian yang relevan berdasarkan perbandingan dengan nilai tengah. Jika nilai tengah lebih kecil dari nilai yang dicari, maka pencarian akan dilanjutkan ke setengah kanan array; sebaliknya, jika nilai tengah lebih besar, pencarian akan berlanjut ke setengah kiri. Proses ini terus diulang hingga data yang dicari ditemukan atau rentang pencarian habis. Kelebihan utama dari Binary Search adalah kompleksitas waktunya yang lebih rendah, yaitu $O(\log n)$, yang menjadikannya sangat efisien untuk pencarian dalam data yang besar.

3. Pencarian pada Data Bertipe 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 field nama atau kategori yang lain selain nim. Solusinya adalah array harus diurutkan berdasarkan nim terlebih dahulu, atau gunakan algoritma sequential search.

GUIDED 1

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("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("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\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:\ALGORITMA PROGRAMING\PraktikumSem2\103112400078_MODUL8> go run "d:\ALGORITMA
File.go"
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 (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
PS D:\ALGORITMA PROGRAMING\PraktikumSem2\103112400078_MODUL8> q

```

DEKSRIPSI:

Program ini melakukan pencarian terhadap elemen dalam array menggunakan dua metode pencarian: pencarian sequential dan pencarian biner. Pada pencarian sequential, program memeriksa setiap elemen dalam array satu per satu tanpa membutuhkan data yang terurut, dan mencatat jumlah langkah yang diperlukan untuk menemukan elemen yang dicari. Setelah itu, array diurutkan, dan pencarian biner dilakukan pada array yang sudah terurut. Pencarian biner lebih efisien karena membagi array menjadi dua bagian pada setiap langkah, mencari elemen di tengah, dan membatasi pencarian pada setengah bagian array yang relevan. Program ini menampilkan langkah-langkah pencarian dan hasil akhir, termasuk jumlah langkah yang dibutuhkan untuk menemukan elemen target atau menyatakan bahwa elemen tidak ditemukan.

GUIDED 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)
    }

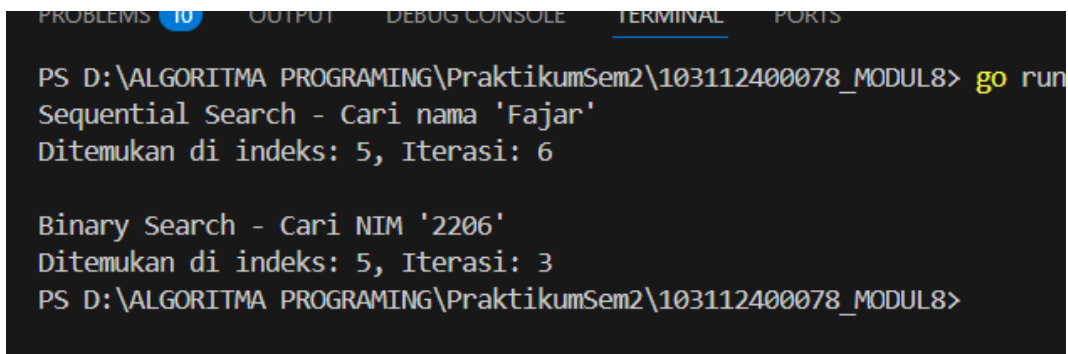
    // Urutkan data berdasarkan NIM untuk binary search
    sort.Slice(data[:n], func(i, j int) bool {
        return data[i].nim < data[j].nim
    })

```

```
// 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:\ALGORITMA PROGRAMING\PraktikumSem2\103112400078_MODUL8> go run
Sequential Search - Cari nama 'Fajar'
Ditemukan di indeks: 5, Iterasi: 6

Binary Search - Cari NIM '2206'
Ditemukan di indeks: 5, Iterasi: 3
PS D:\ALGORITMA PROGRAMING\PraktikumSem2\103112400078_MODUL8>
```

DEKSRIPSI:

Program ini berfungsi untuk mencari data mahasiswa yang terdiri dari nama, NIM, kelas, jurusan, dan IPK menggunakan dua metode pencarian: Sequential Search dan Binary Search. Pada Sequential Search, program mencari mahasiswa berdasarkan nama dengan memeriksa setiap elemen dalam urutan hingga menemukan yang dicari, mencatat jumlah iterasi yang dibutuhkan. Sementara itu, Binary Search digunakan untuk mencari mahasiswa berdasarkan NIM setelah data diurutkan terlebih dahulu. Pencarian biner membagi data menjadi dua bagian di setiap iterasi untuk menemukan NIM yang dicari secara lebih efisien. Program ini menampilkan hasil pencarian beserta jumlah langkah yang dilakukan dalam masing-masing metode pencarian.

II. UNGUIDED

UNGUIDED 1

SOURCE CODE

```
package main
//103112400078 MOHAMMAD REYHAN ARETHA FATIN
import "fmt"

func pilkart() {
    var target int
    var suaraMasuk, suaraSah int = 0, 0
    jumlahVote := make(map[int]int)

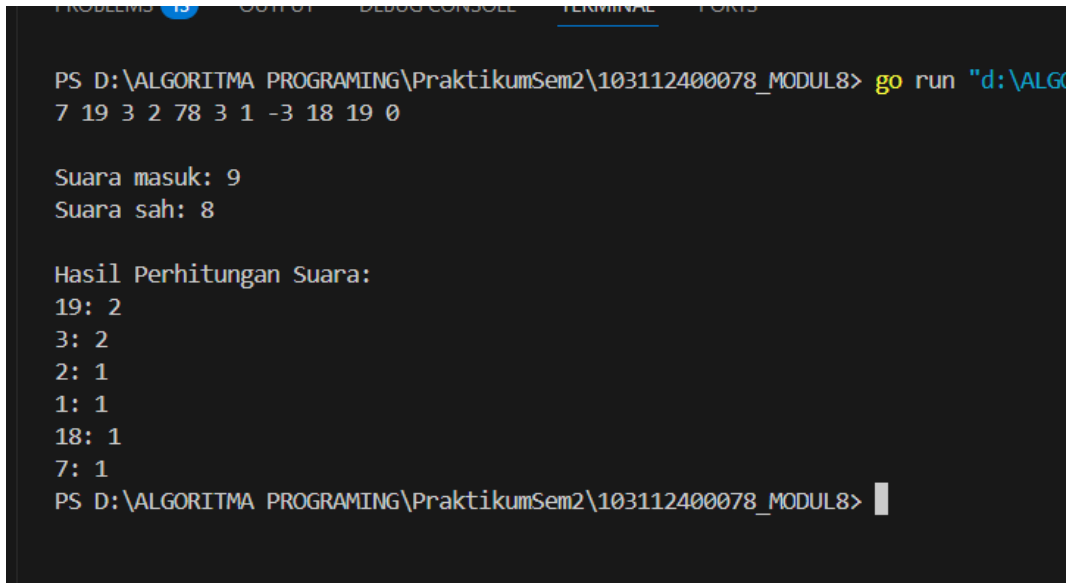
    for {
        _, err := fmt.Scan(&target)
        if err != nil || target < 0 {
            continue
        }
        if target == 0 {
            break
        }
        suaraMasuk++
        if target >= 1 && target <= 20 {
            suaraSah++
            jumlahVote[target]++
        }
    }

    fmt.Printf("\nSuara masuk: %d\n", suaraMasuk)
    fmt.Printf("Suara sah: %d\n", suaraSah)

    fmt.Println("\nHasil Perhitungan Suara:")
    for calon, suara := range jumlahVote {
        fmt.Printf("%d: %d\n", calon, suara)
    }
}

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

OUTPUT

A screenshot of a terminal window showing the execution of a Go program. The terminal has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The command 'go run "d:\ALGORITMA PROGRAMING\PraktikumSem2\103112400078_MODUL8>' is entered. The program outputs a list of numbers: '7 19 3 2 78 3 1 -3 18 19 0'. It then prompts for 'Suara masuk:' (9) and 'Suara sah:' (8). Finally, it displays 'Hasil Perhitungan Suara:' followed by a list of counts for each candidate: '19: 2', '3: 2', '2: 1', '1: 1', '18: 1', and '7: 1'. The terminal prompt returns to 'PS D:\ALGORITMA PROGRAMING\PraktikumSem2\103112400078_MODUL8>'.

```
PS D:\ALGORITMA PROGRAMING\PraktikumSem2\103112400078_MODUL8> go run "d:\ALGORITMA PROGRAMING\PraktikumSem2\103112400078_MODUL8>
7 19 3 2 78 3 1 -3 18 19 0

Suara masuk: 9
Suara sah: 8

Hasil Perhitungan Suara:
19: 2
3: 2
2: 1
1: 1
18: 1
7: 1
PS D:\ALGORITMA PROGRAMING\PraktikumSem2\103112400078_MODUL8>
```

DEKSRIPSI

Program ini mensimulasikan proses pemungutan suara untuk memilih ketua dalam sebuah pemilu, menggunakan angka sebagai identitas calon yang dapat dipilih. Program akan terus menerima input suara hingga pengguna memasukkan angka 0 untuk menandakan akhir pemilihan. Setiap suara yang dimasukkan dihitung sebagai suara masuk, dan jika suara tersebut valid (angka antara 1 hingga 20), maka dihitung sebagai suara sah dan dikaitkan dengan calon yang dipilih. Hasil akhir akan mencakup jumlah suara yang diterima oleh setiap calon serta jumlah total suara yang sah dan tidak sah. Program menggunakan tipe data map untuk menyimpan jumlah suara yang diterima oleh setiap calon, dan mencetak hasil perhitungan suara setelah pemilihan selesai.

UNGUIDED 2

SOURCE CODE

```
package main
// 103112400078 MOHAMMAD REYHAN ARETHA FATIN
import "fmt"
func pilkart() {
    var target, suaraMasuk, suaraSah int
    jumlahVote := make([]int, 20)
    for {
        _, err := fmt.Scan(&target)
        if err != nil || target == 0 {
            break
        }
        if target >= 1 && target <= 20 {
            suaraMasuk++
            suaraSah++
            jumlahVote[target-1]++
        } else {
            suaraMasuk++
        }
    }

    fmt.Println("Suara masuk:", suaraMasuk)
    fmt.Println("Suara sah:", suaraSah)

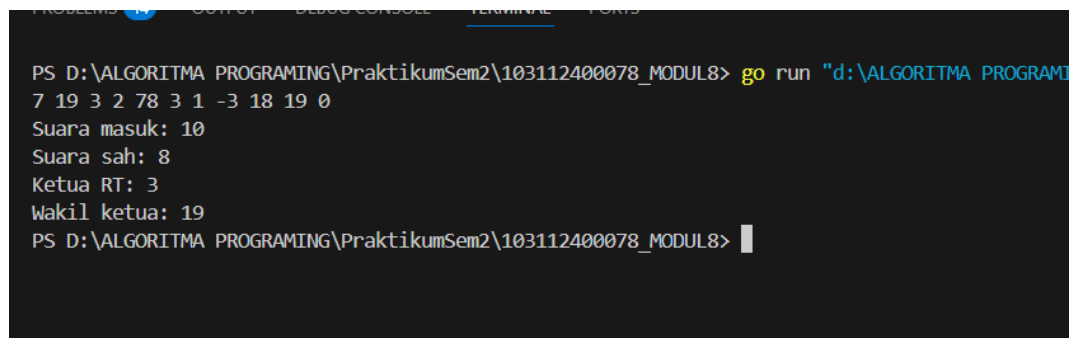
    var ketua, wakilKetua int
    maxVote, secondMaxVote := 0, 0

    for i, vote := range jumlahVote {
        if vote > maxVote {
            secondMaxVote = maxVote
            wakilKetua = ketua
            maxVote = vote
            ketua = i + 1
        } else if vote > secondMaxVote {
            secondMaxVote = vote
            wakilKetua = i + 1
        }
    }

    fmt.Println("Ketua RT:", ketua)
    fmt.Println("Wakil ketua:", wakilKetua)
}
```

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

OUTPUT



```
PS D:\ALGORITMA PROGRAMING\PraktikumSem2\103112400078_MODUL8> go run "d:\ALGORITMA PROGRAMING\PraktikumSem2\103112400078_MODUL8\main.go"  
7 19 3 2 78 3 1 -3 18 19 0  
Suara masuk: 10  
Suara sah: 8  
Ketua RT: 3  
Wakil ketua: 19  
PS D:\ALGORITMA PROGRAMING\PraktikumSem2\103112400078_MODUL8>
```

DEKSRIPSI

Program ini mensimulasikan pemilihan ketua dan wakil ketua untuk suatu organisasi (seperti RT) dengan menggunakan suara yang diberikan oleh peserta. Setiap suara yang dimasukkan akan dihitung, dan jika suara tersebut valid (antara 1 dan 20), maka akan dihitung sebagai suara sah, sementara suara di luar rentang tersebut dianggap tidak sah. Program kemudian akan menghitung jumlah suara yang diterima oleh setiap calon ketua, dan setelah semua suara dihitung, program akan menentukan ketua dan wakil ketua berdasarkan jumlah suara terbanyak. Ketua adalah calon dengan suara terbanyak, sementara wakil ketua adalah calon dengan suara terbanyak kedua. Program juga menampilkan jumlah suara yang masuk, suara sah, serta hasil pemilihan ketua dan wakil ketua.

UNGUIDED 3

SOURCE CODE

```
package main
// 103112400078 MOHAMMAD REYHAN ARETHA FATIN
import "fmt"

func main() {
    var n, k int
    fmt.Scan(&n, &k)
    data := make([]int, n)

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

    sortArray(data)

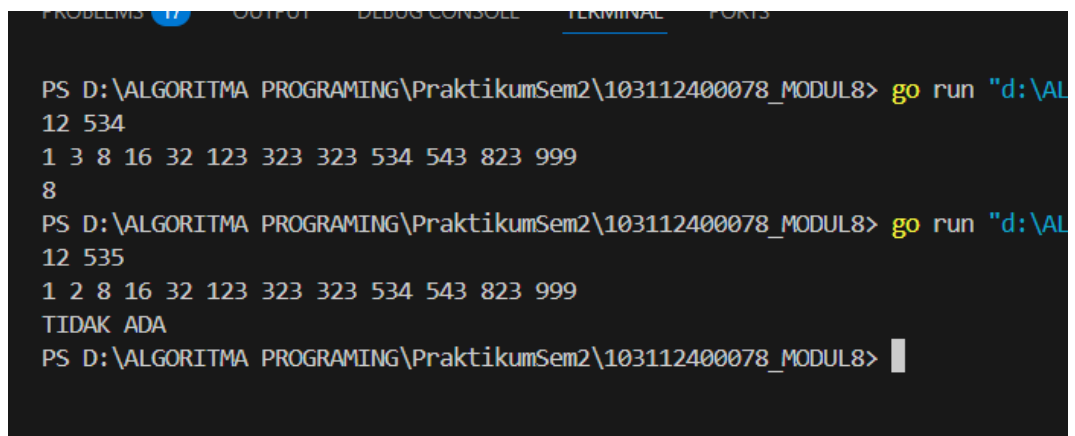
    letak := binarySearch(data, k)
    if letak == -1 {
        fmt.Print("TIDAK ADA")
    } else {
        fmt.Print(letak)
    }
}

func sortArray(arr []int) {
    for i := 0; i < len(arr)-1; i++ {
        for j := 0; j < len(arr)-i-1; j++ {
            if arr[j] > arr[j+1] {
                arr[j], arr[j+1] = arr[j+1], arr[j]
            }
        }
    }
}

func binarySearch(arr []int, x int) int {
    low := 0
    high := len(arr) - 1
    for low <= high {
        mid := (low + high) / 2
        if arr[mid] == x {
            return mid
        } else if arr[mid] < x {
            low = mid + 1
        }
    }
}
```

```
    } else {  
        high = mid - 1  
    }  
}  
return -1  
}
```

OUTPUT



```
PS D:\ALGORITMA PROGRAMING\PraktikumSem2\103112400078_MODUL8> go run "d:\AL  
12 534  
1 3 8 16 32 123 323 323 534 543 823 999  
8  
PS D:\ALGORITMA PROGRAMING\PraktikumSem2\103112400078_MODUL8> go run "d:\AL  
12 535  
1 2 8 16 32 123 323 323 534 543 823 999  
TIDAK ADA  
PS D:\ALGORITMA PROGRAMING\PraktikumSem2\103112400078_MODUL8> |
```

DEKSRIPSI

Program ini melakukan pencarian sebuah angka dalam array menggunakan dua langkah utama: penyortiran dan pencarian biner. Pertama, program menerima input berupa jumlah elemen array dan angka yang ingin dicari. Kemudian, array disortir menggunakan metode Bubble Sort untuk mengurutkan elemen-elemen dari yang terkecil hingga terbesar. Setelah array terurut, program mencari angka yang dimaksud dengan Binary Search, yang secara efisien membagi array menjadi dua bagian untuk menemukan angka tersebut. Jika angka ditemukan, program akan mencetak indeksinya, dan jika tidak ditemukan, program akan menampilkan pesan "TIDAK ADA". Program ini menggabungkan teknik penyortiran dan pencarian yang efisien untuk mencari angka dalam data yang telah diurutkan.

III. KESIMPULAN

Pada praktikum ini, telah mempelajari dan menerapkan dua algoritma pencarian data, yaitu **Sequential Search** dan **Binary Search**, menggunakan bahasa pemrograman Go. Kedua algoritma ini digunakan untuk mencari elemen dalam sebuah array berdasarkan nilai tertentu. **Sequential Search** bekerja dengan memeriksa setiap elemen secara berurutan, dimulai dari elemen pertama hingga elemen terakhir, sampai data yang dicari ditemukan atau pencarian selesai. Di sisi lain, **Binary Search** hanya dapat diterapkan pada data yang sudah terurut dan menggunakan prinsip membandingkan elemen tengah array. Berdasarkan hasil perbandingan tersebut, ruang pencarian akan dipersempit ke bagian kiri atau kanan hingga elemen yang dicari ditemukan. Pemahaman yang mendalam terhadap kedua algoritma ini sangat penting dalam pengolahan data karena memungkinkan pencarian elemen dengan cara yang lebih efisien. Praktikum ini juga memberikan wawasan mengenai perbedaan efisiensi antara algoritma pencarian sederhana dan pencarian yang lebih optimal, khususnya pada struktur data yang sudah terurut.

REFERENSI

MODUL 11. PENCARIAN NILAI ACAK PADA HIMPUNAN
DATA