LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL 8

PENCARIAN NILAI ACAK PADA HIMPUNAN DATA

Oleh:

PRATAMA BINTANG DANISWARA

103112400051

IF-12-01

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

I. DASAR TEORI

- **Sequential Search** (Pencarian Sekuensial) adalah metode pencarian yang dilakukan secara berurutan dari elemen pertama hingga terakhir. Proses berhenti saat data ditemukan atau seluruh elemen telah diperiksa. Metode ini tidak memerlukan data dalam keadaan terurut, namun kurang efisien untuk jumlah data yang besar karena harus mengecek satu per satu.

- **Binary Search** (Pencarian Biner) adalah metode pencarian yang lebih efisien tetapi mengharuskan data sudah **terurut secara ascending**. Algoritma ini bekerja dengan membagi dua rentang data pada setiap langkah: jika nilai tengah lebih kecil dari nilai yang dicari, pencarian dilanjutkan ke bagian kanan; jika lebih besar, ke bagian kiri. Proses ini diulang hingga data ditemukan atau rentang pencarian habis.

II. GUIDED

Source code Guided 1:

```
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\ALPRO\103112400051_MODUL8> go run "d:\Coding\ALPRO\103112400051_MODUL8\103112400051_G1.go"
sequenyial search (data tidak perlu urut):
sequential step 1: cek arr [0]=2
sequential step 2: cek arr [1]=7
sequential step 3: cek arr [2]=9
sequential step 4: cek arr [3]=1
sequential step 5: cek arr [4]=5
sequential step 6: cek arr [5]=6
sequential step 7: cek arr [6]=18
sequential step 8: cek arr [7]=13
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 stop 1: cek arr[4]= 7
binary stop 2: cek arr[6]= 13
Hasil : Ditemukan indeks 6 dalam 2 langkah
PS D:\Coding\ALPRO\103112400051_MODUL8>
```

Deskripsi : Program ini dibuat untuk membandingkan dua cara mencari angka dalam sebuah daftar: **sequential search**, yang mengecek satu per satu tanpa perlu urutan, dan **binary search**, yang lebih cepat tapi butuh data yang sudah diurutkan. Selama program berjalan, kamu bisa melihat proses pencariannya langkah demi langkah, lalu di akhir akan ditunjukkan apakah angkanya ketemu, ada di indeks berapa, dan butuh berapa langkah untuk menemukannya.

Source code Guided 2 :

```
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:\Coding\ALPRO\103112400051_MODUL8> go run "d:\Coding\ALPRO\103
Sequential Search - Cari nama 'Fajar'
Ditemukan di indeks: 5, Iterasi: 6

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

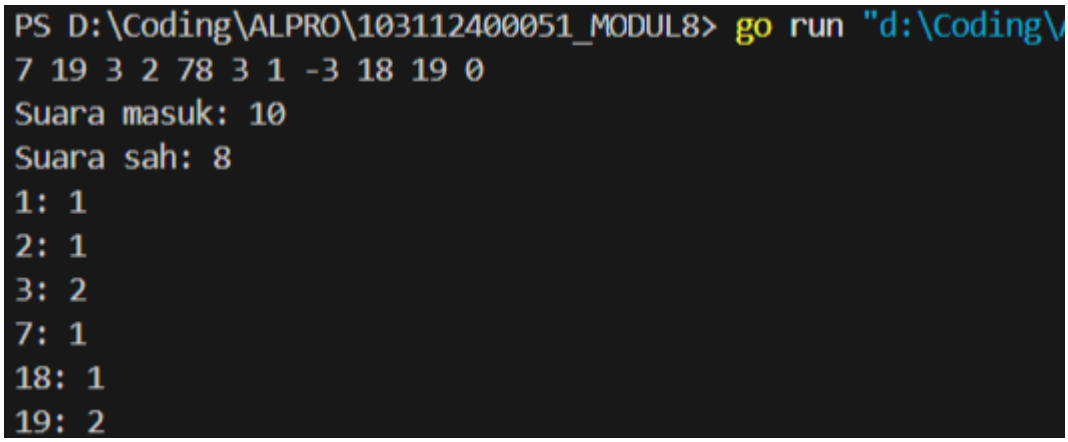
Deskripsi : Program ini dibuat untuk mencari data mahasiswa dengan dua cara berbeda: sequential search dan binary search. Untuk sequential search, program mencari berdasarkan nama dengan memeriksa satu per satu tanpa perlu urutan. Sedangkan binary search mencari berdasarkan NIM, tapi datanya harus diurutkan dulu. Program ini diisi dengan 10 data mahasiswa, lalu mencoba mencari nama "Fajar" dan NIM "2206". Hasil pencarian menunjukkan di indeks berapa data ditemukan dan berapa langkah yang dibutuhkan, jadi kamu bisa lihat mana metode yang lebih efisien.

III. UNGUIDED

Source code Unguided 1 :

```
package main

import "fmt"

func main() {
    var input int
    masuk := 0
    sah := 0
    hasil := make(map[int]int)

    for {
        fmt.Scan(&input)
        masuk++

        if input == 0 {
            masuk--
            break
        }

        if input >= 1 && input <= 20 {
            sah++
            hasil[input]++
        }
    }

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

    for i := 1; i <= 20; i++ {
        if c, e := hasil[i]; e {
            fmt.Printf("%d: %d\n", i, c)
        }
    }
}
```

Output:



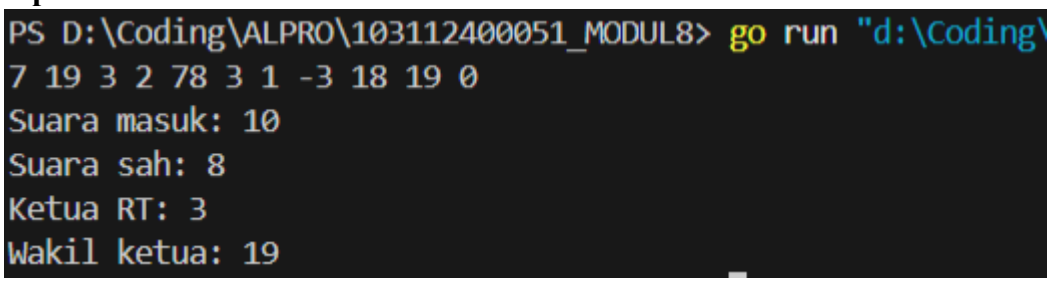
Deskripsi: Program ini dibuat untuk merekap suara dalam sebuah pemungutan suara sederhana. Pengguna memasukkan angka satu per satu sebagai pilihan kandidat (dari nomor 1 sampai 20), dan proses berakhir saat angka 0 dimasukkan. Program akan menghitung berapa total suara yang masuk, berapa yang sah (hanya angka 1–20), dan menampilkan jumlah suara yang didapat oleh tiap kandidat. Jadi, kamu bisa langsung lihat siapa yang dapat suara terbanyak.

Source code Unguided 2 :

```
package main

import "fmt"

func main() {
    var input int
    masuk := 0
    sah := 0
    hasil := make(map[int]int)

    for {
        fmt.Scan(&input)
        masuk++

        if input == 0 {
            masuk--
            break
        }

        if input >= 1 && input <= 20 {
            sah++
            hasil[input]++
        }
    }

    ketua, wakil := cariPemenang(hasil)

    fmt.Printf("Suara masuk: %d\n", masuk)
    fmt.Printf("Suara sah: %d\n", sah)
    fmt.Printf("Ketua RT: %d\n", ketua)
    fmt.Printf("Wakil ketua: %d\n", wakil)
}

func cariPemenang(hasil map[int]int) (int, int) {
    max1, max2 := 0, 0
    ketua, wakil := 0, 0
```

```
    for nomor, suara := range hasil {
        if suara > max1 {
            max2 = max1
            wakil = ketua
            max1 = suara
            ketua = nomor
        } else if suara > max2 && suara < max1 {
            max2 = suara
            wakil = nomor
        } else if suara == max1 {
            if nomor < ketua {
                max2 = max1
                wakil = ketua
                max1 = suara
                ketua = nomor
            } else if nomor < wakil || wakil == 0 {
                wakil = nomor
            }
        } else if suara == max2 {
            if nomor < wakil || wakil == 0 {
                wakil = nomor
            }
        }
    }

    return ketua, wakil
}
```

Output:



Deskripsi: Program Go ini digunakan untuk menghitung hasil pemilihan Ketua dan Wakil Ketua RT berdasarkan suara yang dimasukkan satu per satu oleh pengguna. Input dihentikan saat angka 0 dimasukkan, dan hanya suara sah (bernomor 1 sampai

20) yang akan dihitung. Setelah semua suara masuk, program menentukan siapa yang menjadi Ketua (dengan suara terbanyak) dan Wakil (suara terbanyak kedua). Jika ada jumlah suara yang sama, kandidat dengan nomor lebih kecil akan diprioritaskan. Hasil akhirnya menampilkan total suara masuk, suara sah, serta nomor calon yang terpilih sebagai Ketua dan Wakil Ketua.

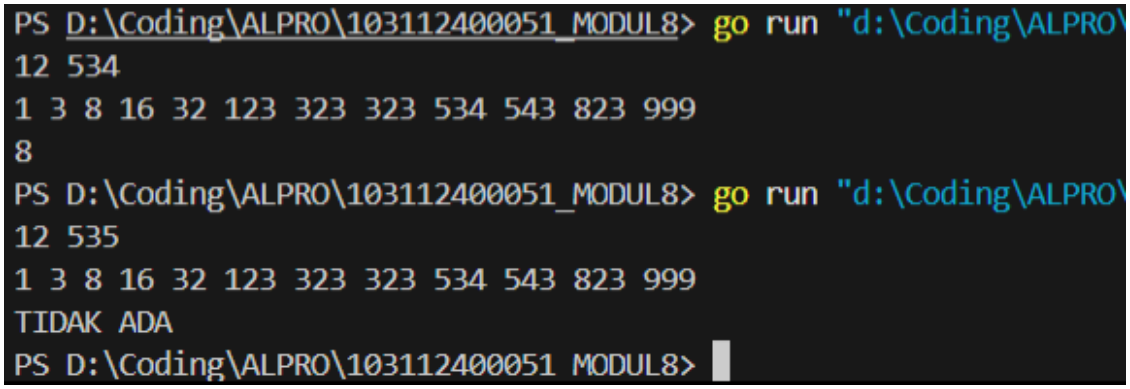
Source code Unguided 3 :

```
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 {
    left := 0
    right := n - 1

    for left <= right {
        mid := left + (right-left)/2

        if data[mid] == k {
            return mid
        }

        if data[mid] < k {
```

```
            left = mid + 1
        } else {
            right = mid - 1
        }
    }

    return -1
}
```

Output :



Deskripsi : Program Go ini digunakan untuk mencari posisi sebuah angka dalam deretan angka yang sudah diurutkan. Pertama, pengguna memasukkan jumlah elemen dan angka yang ingin dicari. Setelah itu, program membaca semua elemen dan menyimpannya dalam array. Proses pencarian dilakukan dengan metode binary search, yaitu membagi array menjadi dua bagian secara terus-menerus hingga menemukan angka yang dicari. Jika ditemukan, program akan mencetak posisi indeksnya; jika tidak, akan ditampilkan pesan "TIDAK ADA".

IV. KESIMPULAN

V. REFERENSI

MODUL 8 Algoritma Pemrograman 2