

**LAPORAN PRAKTIKUM
ALGORITMA PEMROGRAMAN 2
MODUL XI
PENCARIAN NILAI ACAK PADA HIMPUNAN DATA**



Disusun Oleh:

NAMA : Titanio Francy Naddiansa

NIM: 2311102289

KELAS: IF-11-07

**PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO**

2024

I. DASAR TEORI

Pencarian data adalah teknik dasar yang penting dalam pemrograman untuk menemukan elemen tertentu dalam kumpulan data. Salah satu metode dasar adalah Sequential Search, di mana setiap elemen diperiksa satu per satu dari awal hingga akhir. Metode ini sederhana dan cocok untuk data yang tidak terurut, tetapi kurang efisien jika ukuran data besar karena proses pencariannya linear. Untuk data yang sudah terurut, Binary Search menawarkan solusi yang jauh lebih efisien. Algoritma ini bekerja dengan membagi ruang pencarian menjadi dua secara berulang. Dengan membandingkan elemen tengah dengan target, algoritma memutuskan apakah pencarian dilanjutkan di bagian kiri atau kanan. Proses ini secara signifikan mempercepat pencarian dibandingkan dengan metode sequential. Ketika bekerja dengan struktur data yang lebih kompleks, seperti array yang berisi struct, pemilihan metode pencarian tergantung pada kebutuhan. Sequential Search tetap bisa digunakan pada data yang belum terurut, sementara Binary Search membutuhkan data yang sudah diurutkan terlebih dahulu berdasarkan field yang dicari. Pemilihan algoritma pencarian bergantung pada struktur data, ukuran data, urutan data, dan efisiensi yang diinginkan. Setiap metode memiliki keunggulan dan keterbatasan, sehingga memilih algoritma yang sesuai sangat penting untuk memastikan proses pencarian berjalan optimal.

II. GUIDED

1. Source Code

```
package main

import (
    "fmt"
)

// Definisi struct mahasiswa
type mahasiswa struct {
    nama    string
    nim     string
    kelas   string
    jurusan string
    ipk     float64
}

// Definisi array bertipe struct mahasiswa
type arrMhs [2023]mahasiswa

// Sequential Search berdasarkan nama
func SeqSearch_3(T arrMhs, n int, X string) int {
    /* Mengembalikan indeks mahasiswa dengan nama X,
       atau -1 apabila tidak ditemukan pada array T
       yang berisi n data mahasiswa */
    var found int = -1
    var j int = 0
    for j < n && found == -1 {
```

```

        if T[j].nama == X {
            found = j
        }
        j = j + 1
    }
    return found
}

```

// Binary Search berdasarkan nim

```

func BinarySearch_3(T arrMhs, n int, X string) int {
    /* Mengembalikan indeks mahasiswa dengan nim X,
       atau -1 apabila tidak ditemukan pada array T
       yang berisi n data mahasiswa dan terurut membesar berdasarkan nim */
    var found int = -1
    var med int
    var kr int = 0
    var kn int = n - 1

    for kr <= kn && found == -1 {
        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
}

// Fungsi utama untuk demonstrasi
func main() {
    var data arrMhs
    var n int

    // Input jumlah mahasiswa
    fmt.Print("Masukkan jumlah mahasiswa: ")
    fmt.Scanln(&n)

    // Input data mahasiswa
    fmt.Println("Masukkan data mahasiswa:")
    for i := 0; i < n; i++ {
        fmt.Printf("Mahasiswa %d\n", i+1)
        fmt.Print("Nama: ")
        fmt.Scanln(&data[i].nama)
        fmt.Print("NIM: ")
        fmt.Scanln(&data[i].nim)
        fmt.Print("Kelas: ")
        fmt.Scanln(&data[i].kelas)
        fmt.Print("Jurusan: ")
        fmt.Scanln(&data[i].jurusan)
        fmt.Print("IPK: ")
        fmt.Scanln(&data[i].ipk)
    }
}

```

```

// Pencarian berdasarkan nama

var cariNama string

fmt.Print("Masukkan nama mahasiswa yang ingin dicari: ")

fmt.Scanln(&cariNama)

posNama := SeqSearch_3(data, n, cariNama)

if posNama == -1 {
    fmt.Println("Mahasiswa dengan nama tersebut tidak ditemukan.")
} else {
    fmt.Printf("Mahasiswa ditemukan pada indeks: %d\n", posNama)
}

// Pencarian berdasarkan nim

var cariNIM string

fmt.Print("Masukkan NIM mahasiswa yang ingin dicari: ")

fmt.Scanln(&cariNIM)

posNIM := BinarySearch_3(data, n, cariNIM)

if posNIM == -1 {
    fmt.Println("Mahasiswa dengan NIM tersebut tidak ditemukan.")
} else {
    fmt.Printf("Mahasiswa ditemukan pada indeks: %d\n", posNIM)
}
}

```

Screenshoot Output

```
PS C:\Go lang> go run "c:\Go lang\guided9.go"
Masukkan jumlah mahasiswa: 3
Masukkan data mahasiswa:
Mahasiswa 1
Nama: Didik
NIM: 2311102285
Kelas: 07
Jurusan: informatika
IPK: 3.0
Mahasiswa 2
Nama: Alden
NIM: 2311102287
Kelas: 07
Jurusan: informatika
IPK: 2.7
Mahasiswa 3
Nama: Musyafa
NIM: 2311102306
Kelas: 07
Jurusan: informatika
IPK: 3.5
Masukkan nama mahasiswa yang ingin dicari: Didik
Mahasiswa ditemukan pada indeks: 0
Masukkan NIM mahasiswa yang ingin dicari: 2311102287
Mahasiswa ditemukan pada indeks: 1
```

Deskripsi Program

Program ini adalah aplikasi sederhana untuk menyimpan data mahasiswa dan melakukan pencarian menggunakan dua metode: Sequential Search berdasarkan nama mahasiswa dan Binary Search berdasarkan NIM mahasiswa. Data mahasiswa, seperti nama, NIM, kelas, jurusan, dan IPK, disimpan dalam array bertipe struct. Program menerima input jumlah mahasiswa dan data masing-masing mahasiswa, kemudian memungkinkan pengguna untuk mencari mahasiswa berdasarkan nama atau NIM. Pencarian nama menggunakan Sequential Search, yang memeriksa setiap elemen secara berurutan, sementara pencarian NIM menggunakan Binary Search, yang memanfaatkan data yang sudah terurut untuk mempercepat proses pencarian. Hasil pencarian akan menunjukkan indeks lokasi mahasiswa dalam array atau pesan bahwa data tidak ditemukan.

2. Source Code

```
package main

import "fmt"

// Define arrInt as an array of integers with a fixed size
type arrInt [4321]int

// Binary search function for descending sorted array
func BinarySearch_2(T arrInt, n int, X int) int {
    /* Mengembalikan indeks dari X apabila X ditemukan di dalam array T
       yang berisi n buah bilangan bulat terurut secara descending/mencuat,
       atau -1 apabila X tidak ditemukan */
    var found int = -1
    var med int
    var kr int = 0
    var kn int = n - 1

    // Iterasi mencari elemen dengan binary search
    for kr <= kn && found == -1 {
        med = (kr + kn) / 2
        if X > T[med] { // Karena descending, jika X lebih besar, bergerak ke kiri
            kn = med - 1
        } else if X < T[med] { // Jika X lebih kecil, bergerak ke kanan
            kr = med + 1
        } else { // Jika ditemukan
            found = med
        }
    }
```



```

    }

    return found
}

func main() {
    // Contoh penggunaan
    var data arrInt
    var n int

    // Input jumlah elemen
    fmt.Print("Masukkan jumlah elemen dalam array: ")
    fmt.Scanln(&n)

    // Input elemen array (harus terurut secara descending)
    fmt.Println("Masukkan elemen array (harus terurut menurun):")
    for i := 0; i < n; i++ {
        fmt.Printf("Elemen %d: ", i+1)
        fmt.Scanln(&data[i])
    }

    // Input elemen yang dicari
    var search int
    fmt.Print("Masukkan elemen yang ingin dicari: ")
    fmt.Scanln(&search)

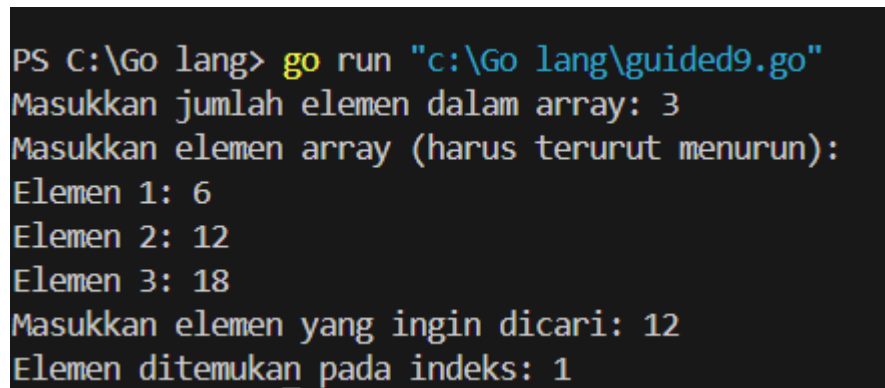
    // Panggil fungsi binary search
    result := BinarySearch_2(data, n, search)

    // Cetak hasil
    if result == -1 {

```

```
    fmt.Println("Elemen tidak ditemukan dalam array.")  
} else {  
    fmt.Printf("Elemen ditemukan pada indeks: %d\n", result)  
}  
}
```

Screenshoot Output



```
PS C:\Go lang> go run "c:\Go lang\guided9.go"  
Masukkan jumlah elemen dalam array: 3  
Masukkan elemen array (harus terurut menurun):  
Elemen 1: 6  
Elemen 2: 12  
Elemen 3: 18  
Masukkan elemen yang ingin dicari: 12  
Elemen ditemukan pada indeks: 1
```

Deskripsi Program

Program ini melakukan pencarian elemen dalam array bilangan bulat yang diurutkan secara menurun (descending) menggunakan metode Binary Search. User memasukkan jumlah elemen dalam array dan elemen-elemen tersebut, yang harus dalam urutan menurun. Program kemudian meminta User untuk memasukkan elemen yang ingin dicari. Dengan Binary Search, program mencari elemen tersebut secara efisien dan mengembalikan indeksnya jika ditemukan, atau menampilkan pesan bahwa elemen tidak ditemukan jika tidak ada dalam array.

3. Source Code

```
package main

import "fmt"

// Define arrStr as an array of strings with a fixed size
type arrStr [1234]string

// Sequential search function
func SeqSearch_1(T arrStr, n int, X string) int {

    /* Mengembalikan indeks dari X apabila X ditemukan di dalam array T
       yang berisi n buah teks, atau -1 apabila X tidak ditemukan */

    var found int = -1

    var j int = 0

    // Iterasi mencari elemen yang cocok
    for j < n && found == -1 {
        if T[j] == X {
            found = j
        }
        j = j + 1
    }

    return found
}

func main() {

    // Contoh penggunaan

    var data arrStr

    var n int

    // Input jumlah elemen

    fmt.Print("Masukkan jumlah elemen dalam array: ")
}
```

```

fmt.Scanln(&n)

// Input elemen array
fmt.Println("Masukkan elemen array:")

for i := 0; i < n; i++ {
    fmt.Printf("Elemen %d: ", i+1)
    fmt.Scanln(&data[i])
}

// Input elemen yang dicari
var search string

fmt.Print("Masukkan elemen yang ingin dicari: ")

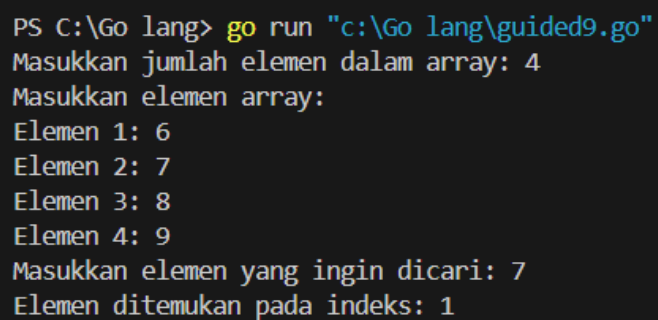
fmt.Scanln(&search)

// Panggil fungsi sequential search
result := SeqSearch_1(data, n, search)

// Cetak hasil
if result == -1 {
    fmt.Println("Elemen tidak ditemukan dalam array.")
} else {
    fmt.Printf("Elemen ditemukan pada indeks: %d\n", result)
}
}

```

Screenshot Output



```

PS C:\Go lang> go run "c:\Go lang\guided9.go"
Masukkan jumlah elemen dalam array: 4
Masukkan elemen array:
Elemen 1: 6
Elemen 2: 7
Elemen 3: 8
Elemen 4: 9
Masukkan elemen yang ingin dicari: 7
Elemen ditemukan pada indeks: 1

```

Deskripsi Program

Program ini melakukan pencarian teks dalam array string menggunakan metode Sequential Search. User memasukkan jumlah elemen dan teks-teks yang akan disimpan dalam array. Setelah itu, program meminta teks yang ingin dicari. Sequential Search memeriksa elemen satu per satu dari awal hingga akhir. Jika teks ditemukan, program mencetak indeksinya; jika tidak, akan menampilkan pesan bahwa teks tersebut tidak ditemukan.

III. UNGUIDED

1. Source Code

```
package main

import (
    "bufio"
    "fmt"
    "os"
    "strconv"
    "strings"
)

func main() {
    scanner := bufio.NewScanner(os.Stdin)

    fmt.Println("Masukkan suara (pisahkan dengan spasi, akhiri dengan 0):")

    scanner.Scan()
    input := scanner.Text()

    data := strings.Split(input, " ")

    totalSuara := 0
    validSuara := 0
    hitungSuara := make(map[int]int)

    for _, item := range data {
        num, err := strconv.Atoi(item)
        if err != nil {
```

```
        continue
    }

    if num == 0 {
        break
    }

    totalSuara++

    if num >= 1 && num <= 20 {
        validSuara++
        hitungSuara[num]++
    }
}

fmt.Printf("Suara masuk: %d\n", totalSuara)
fmt.Printf("Suara sah: %d\n", validSuara)
for i := 1; i <= 20; i++ {
    if count, exists := hitungSuara[i]; exists {
        fmt.Printf("%d: %d\n", i, count)
    }
}
}
```

Screenshoot Output

```
PS C:\Go lang> go run "c:\Go lang\unguided9.go"
Masukkan suara (pisahkan dengan spasi, akhiri dengan 0):
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
```

Deskripsi Program

Program ini menghitung jumlah suara yang valid dan total suara yang dimasukkan User. User diminta untuk memasukkan suara dalam bentuk angka yang dipisahkan dengan spasi, dan mengakhiri input dengan angka 0. Program kemudian memvalidasi suara yang berada di antara 1 hingga 20, menghitung total suara, dan menampilkan jumlah suara yang sah serta frekuensi masing-masing suara yang valid.

2. Source Code

```
package main
```

```
import (
```

```
    "bufio"
```

```
    "fmt"
```

```
    "os"
```

```
    "sort"
```

```
    "strconv"
```

```
    "strings"
```

```
)
```

```
func main() {
```

```
    scanner := bufio.NewScanner(os.Stdin)
```

```
    fmt.Println("Masukkan suara (pisahkan dengan spasi, akhiri dengan 0):")
```

```
    scanner.Scan()
```

```
    input := scanner.Text()
```

```
    data := strings.Split(input, " ")
```

```
    totalSuara := 0
```

```
    validSuara := 0
```

```
    hitungSuara := make(map[int]int)
```

```
    for _, item := range data {
```

```
        num, err := strconv.Atoi(item)
```

```
        if err != nil {
```

```

        continue
    }

    if num == 0 {
        break
    }

    totalSuara++

    if num >= 1 && num <= 20 {
        validSuara++
        hitungSuara[num]++
    }
}

type Kandidat struct {
    Nomor int
    Suara int
}

var daftarKandidat []Kandidat

for nomor, suara := range hitungSuara {
    daftarKandidat = append(daftarKandidat, Kandidat{Nomor: nomor, Suara: suara})
}

sort.Slice(daftarKandidat, func(i, j int) bool {
    if daftarKandidat[i].Suara == daftarKandidat[j].Suara {
        return daftarKandidat[i].Nomor < daftarKandidat[j].Nomor
    }
})

```

```

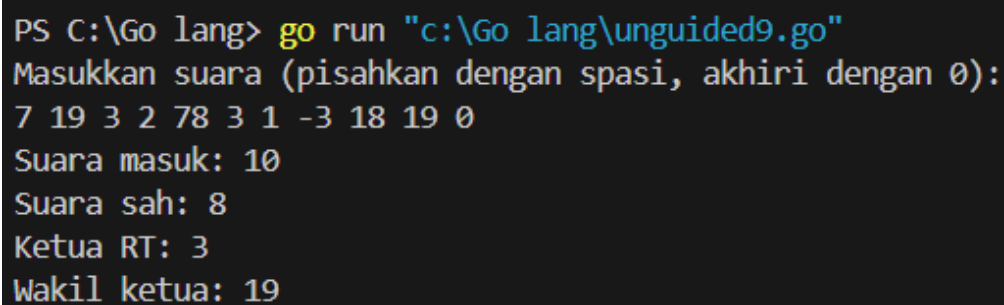
        return daftarKandidat[i].Suara > daftarKandidat[j].Suara
    })

    var ketua, wakil int
    if len(daftarKandidat) > 0 {
        ketua = daftarKandidat[0].Nomor
    }
    if len(daftarKandidat) > 1 {
        wakil = daftarKandidat[1].Nomor
    }

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

```

Screenshoot Output



```

PS C:\Go lang> go run "c:\Go lang\unguided9.go"
Masukkan suara (pisahkan dengan spasi, akhiri dengan 0):
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 menghitung jumlah suara yang valid dan total suara yang dimasukkan User, yang dipisahkan dengan spasi dan diakhiri dengan angka 0. Suara yang sah adalah angka antara 1 hingga 20. Program kemudian mengurutkan kandidat berdasarkan jumlah suara yang diterima, dengan ketentuan jika suara sama, urutan nomor kandidat yang lebih kecil diprioritaskan. Program akan menampilkan jumlah total suara, suara sah, serta nomor kandidat yang terpilih sebagai ketua dan wakil ketua berdasarkan hasil suara terbanyak.

3. Source Code

```
package main

import "fmt"

const NMAX = 1000000

var data [NMAX]int

func main() {
    var n, k int

    fmt.Print("Masukkan jumlah data (n): ")

    fmt.Scan(&n)

    fmt.Print("Masukkan angka yang ingin dicari (k): ")

    fmt.Scan(&k)

    fmt.Println("Masukkan data yang terurut:")

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

    index := binarySearch(n, k)

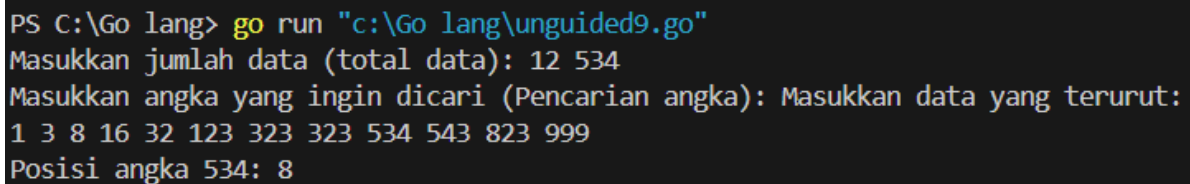
    if index == -1 {
        fmt.Println("TIDAK ADA")
    } else {
        fmt.Printf("Posisi angka %d: %d\n", k, index)
    }
}
```

```

func binarySearch(n, k int) int {
    left, right := 0, n-1
    for left <= right {
        mid := (left + right) / 2
        if data[mid] == k {
            return mid
        } else if data[mid] < k {
            left = mid + 1
        } else {
            right = mid - 1
        }
    }
    return -1
}

```

Screenshoot Output



```

PS C:\Go lang> go run "c:\Go lang\unguided9.go"
Masukkan jumlah data (total data): 12 534
Masukkan angka yang ingin dicari (Pencarian angka): Masukkan data yang terurut:
1 3 8 16 32 123 323 323 534 543 823 999
Posisi angka 534: 8

```

Deskripsi Program

Program ini mengimplementasikan pencarian angka dalam array yang terurut menggunakan algoritma Binary Search. User diminta untuk memasukkan jumlah data dan angka yang ingin dicari. Program akan mencari angka tersebut dalam array yang telah diinput. Jika angka ditemukan, program akan menampilkan posisi indeksinya; jika tidak, program akan mencetak "TIDAK ADA". Algoritma Binary Search digunakan untuk efisiensi pencarian dengan membagi ruang pencarian secara berulang.