

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



Oleh:

NAMA: Didik Weka Pratama

NIM: 2311102285

KELAS: S1 IF-11-07

**S1 TEKNIK INFORMATIKA  
INSTITUT TEKNOLOGI TELKOM PURWOKERTO  
2024**

## **I. DASAR TEORI**

### **Algoritma Binary Search**

Binary Search adalah algoritma pencarian yang bekerja dengan cara membagi data menjadi dua bagian secara berulang dan memilih bagian yang kemungkinan besar mengandung elemen yang dicari. Langkah-langkah umum dari algoritma Binary Search adalah sebagai berikut:

1. Tentukan indeks kiri (left) dan kanan (right) dari data.
2. Cari elemen tengah (mid), yaitu elemen pada posisi  $(\text{left} + \text{right}) / 2$ .
3. Jika elemen tengah adalah elemen yang dicari, kembalikan posisinya.
4. Jika elemen yang dicari lebih kecil dari elemen tengah, cari pada bagian kiri (bawah).
5. Jika elemen yang dicari lebih besar dari elemen tengah, cari pada bagian kanan (atas).
6. Ulangi langkah ini hingga elemen ditemukan atau rentang pencarian (left dan right) tidak valid lagi.

Keunggulan dari Binary Search adalah kemampuannya untuk mencari data dengan cepat pada dataset yang sudah terurut, dibandingkan dengan pencarian linear yang memerlukan waktu  $O(n)$ .

### **Struktur Data Array**

Pada penerapan Binary Search, struktur data yang digunakan adalah array. Array adalah jenis struktur data yang menyimpan elemen-elemen secara berurutan dalam memori dan memiliki indeks yang dapat diakses. Keuntungan menggunakan array adalah elemen-elemen yang disimpan dapat diakses secara langsung menggunakan indeks, dengan waktu akses konstan  $O(1)$ .

Namun, kelemahan utama dari array adalah ukurannya yang tetap setelah deklarasi, yang berarti kita tidak dapat menambah atau mengurangi ukuran array setelah array dibuat. Oleh karena itu, pada program ini, array digunakan untuk menyimpan data yang sudah terurut, dan Binary Search diterapkan untuk mencari posisi elemen yang dicari.

## 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.Println("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.Println("Nama: ")
        fmt.Scanln(&data[i].nama)
        fmt.Println("NIM: ")
        fmt.Scanln(&data[i].nim)
        fmt.Println("Kelas: ")
        fmt.Scanln(&data[i].kelas)
        fmt.Println("Jurusan: ")
        fmt.Scanln(&data[i].jurusan)
        fmt.Println("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)
    }
}

```

## **Screenshot Output**

```

PS C:\golang> go run "c:\golang\src\hello.go"
Masukkan jumlah mahasiswa: 3
Masukkan data mahasiswa:
Mahasiswa 1
Nama: Didik
NIM: 2311102285
Kelas: IF-11-07
Jurusan: INFORMATIK
IPK: 3.9
Mahasiswa 2
Nama: Titan
NIM: 2311102289
Kelas: IF-11-07
Jurusan: Informatik
IPK: 4.0
Mahasiswa 3
Nama: ARIF
NIM: 231110300
Kelas: IF-11-08
Jurusan: Informatik
IPK: 3.8
Masukkan nama mahasiswa yang ingin dicari: titan
Mahasiswa dengan nama tersebut tidak ditemukan.
Masukkan NIM mahasiswa yang ingin dicari: 2311102289
Mahasiswa ditemukan pada indeks: 1
PS C:\golang>

```

## Deskripsi Program

Program ini mengelola data mahasiswa dan melakukan pencarian berdasarkan nama atau NIM. Data mahasiswa disimpan dalam array bertipe struct mahasiswa, yang mencakup informasi seperti nama, NIM, kelas, jurusan, dan IPK. Program memungkinkan pengguna untuk memasukkan sejumlah data mahasiswa, lalu melakukan dua jenis pencarian:

1. **Sequential Search** untuk mencari mahasiswa berdasarkan nama, dengan memeriksa setiap elemen array satu per satu hingga ditemukan kecocokan atau seluruh data selesai diperiksa.
2. **Binary Search** untuk mencari mahasiswa berdasarkan NIM, yang memanfaatkan array yang sudah terurut untuk mempercepat pencarian melalui teknik pembagian (divide and conquer).

Hasil pencarian menampilkan indeks mahasiswa dalam array jika ditemukan, atau pesan kesalahan jika data tidak ditemukan. Program ini menunjukkan implementasi struktur data, algoritma pencarian, dan interaksi input/output dalam Go.

## 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/menciut,
       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 {

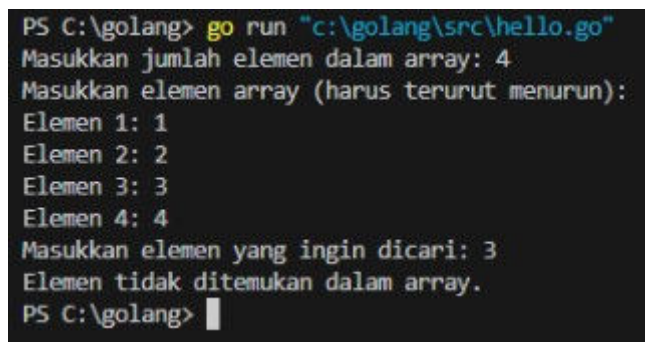
    fmt.Printf("Elemen ditemukan pada indeks: %d\n", result)

}

}

```

## SCREENSHOOT OUTPUT



```

PS C:\golang> go run "c:\golang\src\hello.go"
Masukkan jumlah elemen dalam array: 4
Masukkan elemen array (harus terurut menurun):
Elemen 1: 1
Elemen 2: 2
Elemen 3: 3
Elemen 4: 4
Masukkan elemen yang ingin dicari: 3
Elemen tidak ditemukan dalam array.
PS C:\golang>

```

## Deskripsi program

Program ini mencari sebuah elemen dalam array bilangan bulat yang terurut secara menurun (descending) menggunakan algoritma **Binary Search**. Program meminta pengguna untuk memasukkan jumlah elemen array, kemudian memasukkan elemen-elemen tersebut yang harus diurutkan secara menurun. Pengguna juga diminta memasukkan elemen yang ingin dicari. Fungsi `BinarySearch_2` digunakan untuk mencari elemen tersebut dengan prinsip pembagian (divide and conquer), di mana array terus dibagi dua hingga elemen ditemukan atau pencarian selesai. Jika elemen ditemukan, program akan menampilkan indeks elemen tersebut; jika tidak, akan muncul pesan bahwa elemen tidak ditemukan. Program ini mengilustrasikan penggunaan algoritma pencarian yang efisien untuk array terurut.

## 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)

}

}

```

## SCREENSHOOT OUTPUT

```
PS C:\golang> go run "c:\golang\src\hello.go"
Masukkan jumlah elemen dalam array: 4
Masukkan elemen array:
Elemen 1: 12
Elemen 2: 14
Elemen 3: 13
Elemen 4: 11
Masukkan elemen yang ingin dicari: 11
Elemen ditemukan pada indeks: 3
PS C:\golang> █
```

## Deskripsi program

Program ini melakukan pencarian elemen berupa teks dalam sebuah array menggunakan algoritma **Sequential Search**. Pengguna diminta memasukkan jumlah elemen array, diikuti dengan pengisian elemen-elemen array berupa teks. Selanjutnya, pengguna memasukkan elemen teks yang ingin dicari. Fungsi `SeqSearch_1` melakukan pencarian dengan memeriksa elemen array satu per satu dari awal hingga akhir. Jika elemen yang dicari ditemukan, program akan menampilkan indeksnya; jika tidak, program akan menampilkan pesan bahwa elemen tidak ditemukan. Program ini cocok untuk pencarian pada array dengan elemen tidak terurut, meskipun efisiensinya linear karena memeriksa setiap elemen secara berurutan.

### III. UNGUIDED

#### No 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 {
```

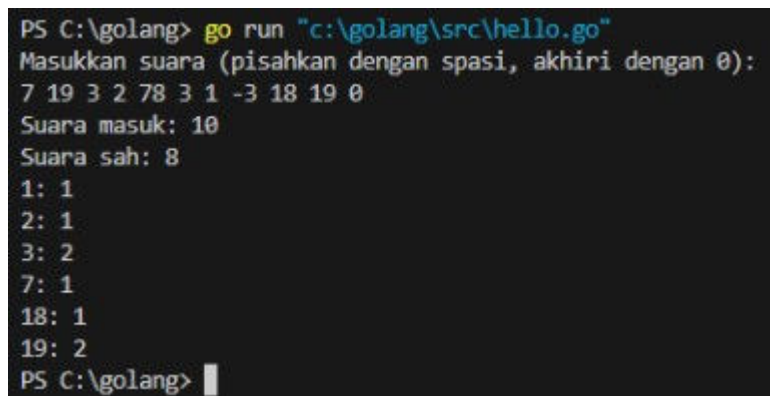
```

        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:\golang> go run "c:\golang\src\hello.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
PS C:\golang>

```

### Deskripsi Program

Program ini menghitung hasil suara dalam suatu pemilihan, termasuk total suara yang masuk, jumlah suara yang sah, dan distribusi suara untuk kandidat bernomor 1 hingga 20. Pengguna diminta memasukkan data suara yang dipisahkan dengan spasi, diakhiri dengan angka 0 sebagai penanda akhir input. Program menggunakan map untuk menghitung frekuensi suara sah berdasarkan nomor kandidat. Hanya suara dengan nomor kandidat 1 hingga 20 yang dianggap sah. Data suara yang tidak valid atau di luar rentang tersebut diabaikan. Program kemudian menampilkan total suara masuk, jumlah suara sah, dan jumlah suara yang diperoleh masing-masing kandidat yang mendapat suara.

### No 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})
}

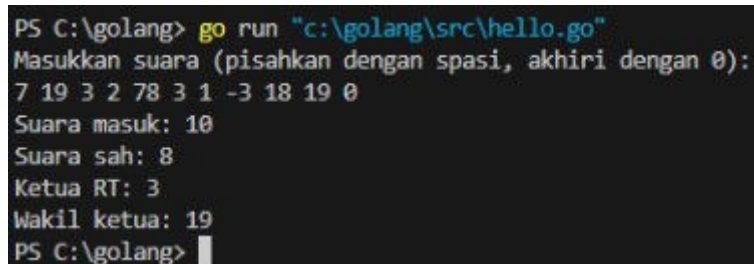
sort.Slice(daftarKandidat, func(i, j int) bool {
    if daftarKandidat[i].Suara == daftarKandidat[j].Suara {
        return daftarKandidat[i].Nomor < daftarKandidat[j].Nomor
    }
    return daftarKandidat[i].Suara > daftarKandidat[j].Suara
}))

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:\golang> go run "c:\golang\src\hello.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
PS C:\golang>

```

### Deskripsi Program



Program ini bertujuan menghitung hasil pemilihan suara untuk memilih ketua dan wakil ketua RT berdasarkan nomor kandidat 1 hingga 20. Pengguna memasukkan data suara yang dipisahkan dengan spasi, diakhiri dengan angka 0 sebagai penanda akhir input. Program menghitung total suara masuk, jumlah suara sah, serta distribusi suara untuk masing-masing kandidat menggunakan struktur data map. Selanjutnya, data kandidat dan jumlah suara mereka disusun dalam slice yang diurutkan secara menurun berdasarkan jumlah suara (dengan prioritas nomor kandidat terkecil jika suara sama). Kandidat dengan suara terbanyak ditetapkan sebagai ketua, dan kandidat dengan suara terbanyak kedua ditetapkan sebagai wakil ketua. Hasil akhirnya mencakup total suara masuk, suara sah, nomor ketua RT, dan nomor wakil ketua RT.

### No 3. Source Code

```
package main

import "fmt"

const MAX_SIZE = 1000000

var numbers [MAX_SIZE]int

func main() {
    var totalNumbers, targetNumber int
    fmt.Print("Masukkan jumlah data (totalNumbers): ")
    fmt.Scan(&totalNumbers)
    fmt.Print("Masukkan angka yang ingin dicari (targetNumber): ")
    fmt.Scan(&targetNumber)

    fmt.Println("Masukkan data yang terurut:")
    for i := 0; i < totalNumbers; i++ {
        fmt.Scan(&numbers[i])
    }

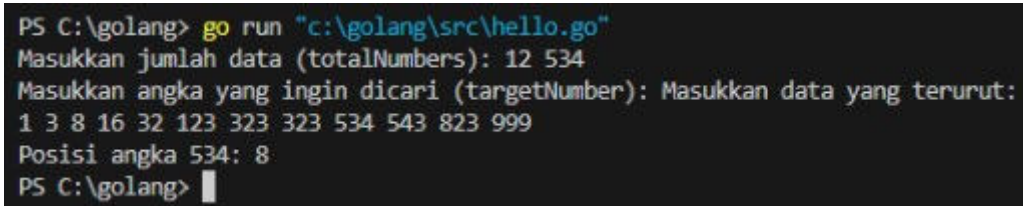
    position := binarySearch(totalNumbers, targetNumber)
    if position == -1 {
        fmt.Println("TIDAK ADA")
    } else {
        fmt.Printf("Posisi angka %d: %d\n", targetNumber, position)
    }
}
```

```

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

```

### Screenshoot Output



```

PS C:\golang> go run "c:\golang\src\hello.go"
Masukkan jumlah data (totalNumbers): 12 534
Masukkan angka yang ingin dicari (targetNumber): Masukkan data yang terurut:
1 3 8 16 32 123 323 323 534 543 823 999
Posisi angka 534: 8
PS C:\golang>

```

### Deskripsi Program

Program ini mencari posisi sebuah angka dalam kumpulan data yang terurut menggunakan algoritma **Binary Search**, yang efisien dengan kompleksitas waktu **O(log n)**. Pengguna diminta memasukkan jumlah data (totalNumbers), angka yang ingin dicari (targetNumber), serta data angka yang terurut dalam array. Fungsi binarySearch melakukan pencarian dengan prinsip pembagian (divide and conquer), di mana array dibagi dua secara iteratif hingga elemen ditemukan atau proses pencarian selesai. Jika angka yang dicari ditemukan, program menampilkan posisinya (indeks). Jika tidak ditemukan, program menampilkan pesan "TIDAK ADA". Program ini cocok untuk pencarian cepat dalam data yang sudah terurut.