

**LAPORAN PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL 8**



**DISUSUN OLEH:  
RIZKINA AZIZAH  
103112400082  
S1 IF-12-01**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA  
FAKULTAS INFORMATIKA  
TELKOM UNIVERSITY PURWOKERTO  
2025**

## DASAR TEORI

### A. Binary Search Algorithm

Algoritma pencarian yang digunakan dalam array yang diurutkan dengan membagi interval pencarian menjadi dua secara berulang. Ide pencarian biner adalah menggunakan informasi bahwa array diurutkan dan mengurangi kompleksitas waktu menjadi  $O(\log N)$ .

### B. Kondisi untuk menerapkan Algoritma Pencarian Biner dalam Struktur Data

Untuk menerapkan algoritma Pencarian Biner:

- Struktur data harus diurutkan.
- Akses ke elemen mana pun dari struktur data harus memerlukan waktu yang konstan.

### C. Algoritma Pencarian Biner

Berikut adalah algoritma langkah demi langkah untuk Pencarian Biner:

- Membagi ruang pencarian menjadi dua bagian dengan **mencari indeks tengah “mid”**.
- Bandingkan elemen tengah ruang pencarian dengan **kunci**.
- Jika **kunci** ditemukan di elemen tengah, proses dihentikan.
- Jika **kunci** tidak ditemukan di elemen tengah, pilih bagian mana yang akan digunakan sebagai ruang pencarian berikutnya.
  - Jika **kuncinya** lebih kecil dari elemen tengah, maka sisi **kiri** digunakan untuk pencarian berikutnya.
  - Jika **kuncinya** lebih besar dari elemen tengah, maka sisi **kanan** digunakan untuk pencarian berikutnya.
- Proses ini dilanjutkan hingga **kunci** ditemukan atau total ruang pencarian habis.

Bagaimana Algoritma Pencarian Biner bekerja?

Untuk memahami cara kerja pencarian biner, perhatikan ilustrasi berikut:

Pertimbangkan sebuah array  $arr[] = \{2, 5, 8, 12, 16, 23, 38, 56, 72, 91\}$ , dan  $target = 23$ .

### D. Bagaimana Menerapkan Algoritma Pencarian Biner?

Algoritma **Pencarian Biner** dapat diimplementasikan dengan dua cara berikut

- Algoritma Pencarian Biner Iteratif
- Algoritma Pencarian Biner Rekursif

## GUIDED

### 1. 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("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() {
    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)
    }
}

```

#### Deskripsi Program:

- Program ini digunakan untuk mencari data dalam array menggunakan dua metode: sequential search dan binary search.
- Fungsi *sequentialSearch*, yang mencari data satu per satu dari awal hingga akhir tanpa perlu mengurutkan data terlebih dahulu. Berfungsi sebagai:
  - Melakukan iterasi pada setiap elemen array.
  - Mencetak langkah pengecekan setiap elemen.
  - Mengembalikan indeks jika ditemukan, serta jumlah langkah iterasi.
- Fungsi *binarySearch* yang memerlukan data dalam kondisi terurut dan mencari target dengan cara membagi dua bagian data pada setiap langkahnya. Berfungsi sebagai :
  - Melakukan pencarian secara efisien dengan membagi rentang array secara berulang.
  - Mencetak langkah pengecekan elemen tengah pada setiap iterasi.
  - Mengembalikan indeks jika ditemukan, serta jumlah langkah iterasi.
- Fungsi *main* berfungsi untuk
  - Mendefinisikan array data dan nilai target yang ingin dicari.
  - Melakukan pencarian menggunakan *sequentialSearch* tanpa mengurutkan data.
  - Menampilkan hasil pencarian dan jumlah langkah.
  - Mengurutkan data menggunakan *sort.Float64s*.
  - Menampilkan data yang telah terurut.
  - Melakukan pencarian menggunakan *binarySearch*
  - Menampilkan hasil pencarian dan jumlah langkah.

## 2. 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)
}
}

```

#### Deskripsi Program:

- Program ini digunakan untuk mencari data mahasiswa menggunakan dua metode pencarian: Sequential Search dan Binary Search.
- Data mahasiswa disimpan dalam array bertipe *arrMhs* yang berisi elemen bertipe *struct mahasiswa* (dengan atribut nama, NIM, kelas, jurusan, dan IPK).



- *func SeqSearch\_3* berfungsi untuk melakukan pencarian nama mahasiswa menggunakan metode Sequential Search (tidak perlu data terurut).
- *func BinarySearch\_3* berfungsi untuk melakukan pencarian berdasarkan NIM menggunakan metode Binary Search, sehingga data harus sudah terurut berdasarkan NIM terlebih dahulu.
- *func main* berfungsi untuk:
  - Mengisi data mahasiswa secara manual.
  - Melakukan pencarian nama menggunakan *SeqSearch\_3* dan menampilkan hasilnya.
  - Mengurutkan data berdasarkan NIM menggunakan *sort.Slice*
  - Melakukan pencarian NIM menggunakan *BinarySearch\_3* dan menampilkan hasilnya.

## UNGUIDED

### 1. Unguided 1

#### Source Code:

```
package main
import "fmt"
func main() {
    var n, tot, sah int
    c := make(map[int]int)

    for {
        fmt.Scan(&n)
        if n == 0 {
            break
        }
        tot++
        if n >= 1 && n <= 20 {
            sah++
            c[n]++
        }
    }
}
```

```

    }
}
fmt.Println("Suara masuk:", tot)
fmt.Println("Suara sah:", sah)
for i := 1; i <= 20; i++ {
    if c[i] > 0 {
        fmt.Printf("%d: %d\n", i, c[i])
    }
}
}
}

```

Output:

```

PS C:\Users\HP\Documents\ittp\TUP smst 2\alpro smt 2> go run
"c:\Users\HP\Documents\ittp\TUP smst 2\alpro smt 2\1031124000
82_MODUL8\Unguided\103112400082_Unguided1.go"
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 digunakan untuk menghitung dan merekap jumlah suara masuk dan suara sah dari input angka.
- Input berupa angka bulat (n), diulang terus hingga memasukkan angka 0 sebagai tanda berhenti.
- Setiap angka yang dimasukkan akan dihitung sebagai suara masuk (tot).
- Jika angka berada dalam rentang 1 sampai 20, maka dianggap sebagai suara sah dan jumlahnya disimpan dalam map c.
- Output :
  - Program menampilkan total suara masuk (tot) dan suara sah (sah).
  - Menampilkan rekap jumlah suara sah untuk setiap kandidat (1–20) yang mendapatkan suara.

## 2. Latihan 2

### Source Code:

```
package main

import "fmt"

func main() {
    var s, total, sah int
    suara := make(map[int]int)

    for {
        fmt.Scan(&s)
        if s == 0 {
            break
        }
        total++
        if s >= 1 && s <= 20 {
            sah++
            suara[s]++
        }
    }

    fmt.Println("Suara masuk:", total)
    fmt.Println("Suara sah:", sah)

    var ketua, wakil int
    max1, max2 := 0, 0

    for i := 1; i <= 20; i++ {
        skrg := suara[i]
        if skrg > max1 {
            max2 = max1
            max1 = skrg
        }
    }
}
```

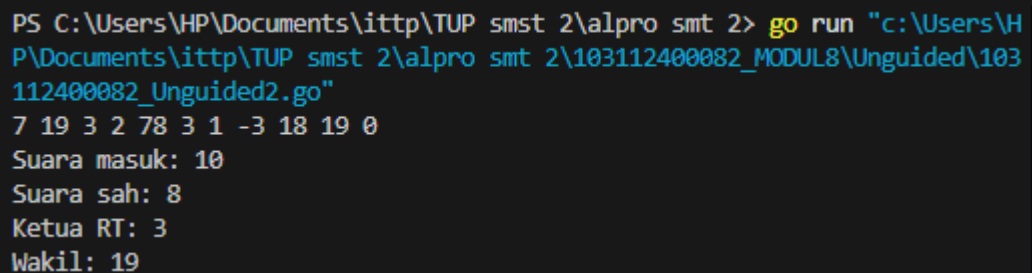
```

        ketua = i
    } else if skrg > max2 && i != ketua {
        max2 = skrg
        wakil = i
    }
}

fmt.Printf("Ketua RT: %d\n", ketua)
if max2 > 0 {
    fmt.Printf("Wakil: %d\n", wakil)
} else {
    fmt.Println("Wakil: Tidak ada")
}
}

```

Output:



```

PS C:\Users\HP\Documents\ittp\TUP smst 2\alpro smt 2> go run "c:\Users\HP\Documents\ittp\TUP smst 2\alpro smt 2\103112400082_MODUL8\Unguided\103112400082_Unguided2.go"
7 19 3 2 78 3 1 -3 18 19 0
Suara masuk: 10
Suara sah: 8
Ketua RT: 3
Wakil: 19

```

Deskripsi program:

- Program ini digunakan untuk merekap hasil pemilihan Ketua dan Wakil RT berdasarkan suara yang dimasukkan.
- Input berupa angka (nomor kandidat) dan akan berhenti saat pengguna memasukkan angka 0.
- Setiap angka yang dimasukkan dihitung sebagai suara masuk (total).
- Jika angka berada dalam rentang 1 sampai 20, dihitung sebagai suara sah dan dicatat dalam map suara.
- Output:
  - Menampilkan jumlah total suara masuk dan suara sah.
  - Menentukan Ketua RT sebagai kandidat dengan suara terbanyak.

- Menentukan Wakil RT sebagai kandidat dengan suara terbanyak kedua (selain ketua).
- Jika hanya ada satu kandidat yang mendapat suara, maka wakil tidak ditampilkan.

### 3. Latihan 3

#### Source Code:

```
package main

import "fmt"

const MAKS = 1000000
var arr [MAKS]int

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

func cari(n, x int) int {
    kiri, kanan := 0, n-1
    for kiri <= kanan {
        tengah := (kiri + kanan) / 2
        if arr[tengah] == x {
            return tengah
        } else if arr[tengah] < x {
            kiri = tengah + 1
        } else {
            kanan = tengah - 1
        }
    }
    return -1
}
```

```

}

func main() {
    var n, x int
    fmt.Scan(&n, &x)
    isiArray(n)
    pos := cari(n, x)

    if pos != -1 {
        fmt.Println(pos)
    } else {
        fmt.Println("TIDAK ADA")
    }
}

```

Output:

```

PS C:\Users\HP\Documents\ittp\TUP smst 2\alpro smt 2> go run "c:\Users\HP\
Documents\ittp\TUP smst 2\alpro smt 2\103112400082_MODUL8\Unguided\1031124
00082_Unguided3.go"
12 534
1 3 8 16 32 123 323 323 534 543 823 999
8
PS C:\Users\HP\Documents\ittp\TUP smst 2\alpro smt 2> go run "c:\Users\HP\
Documents\ittp\TUP smst 2\alpro smt 2\103112400082_MODUL8\Unguided\1031124
00082_Unguided3.go"
12 535
1 3 8 16 32 123 323 323 534 543 823 999
TIDAK ADA

```

Deskripsi Program:

- Program ini digunakan untuk mencari posisi suatu nilai dalam array terurut menggunakan metode Binary Search.
- Konstanta *MAKS* menentukan ukuran maksimum array (1.000.000).
- Fungsi *isiArray(n)* digunakan untuk mengisi array dengan n buah elemen dari input.
- Fungsi *cari(n, x)* menerapkan algoritma Binary Search untuk mencari nilai x dalam array yang sudah terurut:
  - Jika ditemukan, mengembalikan indeks posisi x.

- Jika tidak ditemukan, mengembalikan -1.
- Fungsi *main* berfungsi untuk:
  - Membaca nilai n (jumlah data) dan x (nilai yang dicari).
  - Mengisi array.
  - Mencari nilai x menggunakan fungsi cari.
  - Menampilkan posisi jika ditemukan, atau "TIDAK ADA" jika tidak ditemukan.

## DAFTAR PUSTAKA

PC PRATHEESH(2023), Menjelajahi Algoritma Penyortiran dan Pencarian dengan Golang —  
Pencarian Biner (<https://pcpratheesh.medium.com/exploring-sorting-and-searching-algorithms-with-golang-binary-search-6679716bcd3a> ,2023)