

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



Oleh:

NAMA : SETYO NUGROHO

NIM : 103112400024

KELAS : S1IF-12-01

**S1 TEKNIK INFORMATIKA  
TELKOM UNIVERSITY PURWOKERTO**

**2025**

## I. DASAR TEORI

Pencarian nilai acak pada himpunan data merupakan proses pemilihan elemen dari suatu kumpulan data secara tidak terprediksi dan tidak bergantung pada urutan atau pola tertentu. Proses ini banyak dimanfaatkan dalam berbagai bidang ilmu komputer, seperti kecerdasan buatan, statistik komputasi, pemodelan simulasi, dan pengembangan perangkat lunak, khususnya dalam pengambilan sampel, pengacakan data, ataupun pengujian.

Dalam konteks bahasa pemrograman Go, pencarian nilai acak pada himpunan data dilakukan dengan menggunakan pendekatan pseudorandom number generation, yaitu angka acak yang dihasilkan melalui algoritma deterministik namun dengan hasil yang menyerupai keacakan sejati. Go menyediakan pustaka standar bernama `math/rand` yang digunakan untuk mengakses fungsi-fungsi pembangkit angka acak.

Secara umum, proses pencarian nilai acak dalam sebuah himpunan data dalam Go mencakup beberapa tahapan berikut:

1. **Representasi Data dalam Struktur yang Sesuai**

Data yang akan dipilih secara acak harus disimpan terlebih dahulu dalam suatu struktur data seperti array atau slice, yang memungkinkan akses elemen berdasarkan indeks.

2. **Inisialisasi Pembangkit Angka Acak (Seed)**

Untuk menghasilkan nilai acak yang bervariasi pada setiap eksekusi program, diperlukan proses inisialisasi atau *seeding* terhadap pembangkit angka acak. Dalam Go, nilai seed yang umum digunakan berasal dari waktu sistem, sehingga hasil acakan tidak bersifat statis.

3. **Pemilihan Indeks Secara Acak**

Setelah data tersedia dan pembangkit angka acak diinisialisasi, program dapat menentukan indeks secara acak dari rentang indeks yang valid dalam himpunan data tersebut. Indeks ini digunakan untuk mengakses elemen tertentu dari kumpulan data.

4. **Pengambilan Nilai Berdasarkan Indeks Acak**

Indeks acak yang diperoleh kemudian digunakan untuk mengambil nilai dari himpunan data, sehingga nilai yang diambil merupakan hasil dari proses acak.

Pendekatan ini sangat berguna dalam berbagai skenario praktis seperti pemilihan elemen secara acak dalam game, pengacakan urutan soal dalam aplikasi kuis, atau dalam proses pengambilan sampel data pada eksperimen ilmiah.

Adapun keandalan hasil acak sangat bergantung pada kualitas algoritma pembangkit bilangan acak dan teknik seeding yang digunakan. Oleh karena itu, penggunaan *seed* yang dinamis (misalnya berdasarkan waktu) sangat dianjurkan untuk menjamin variasi hasil dalam setiap eksekusi program.

Dengan demikian, pencarian nilai acak pada himpunan data dalam bahasa Go merupakan kombinasi antara representasi data yang efisien, penggunaan pustaka acak yang tepat, dan penerapan konsep dasar probabilitas untuk menghasilkan pilihan yang tidak dapat diprediksi sebelumnya.

## II. GUIDED

### 1

#### Source code:

```
package main

// 103112400024 Setyo Nugroho

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("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:\Praktikum\semester2\Modul 8> go run "d:\Praktikum\semester2\Modul 8\G1\G1.go"
Sequential Search (data tidak perlu urut)
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:\Praktikum\semester2\Modul 8>

```

### Deskripsi:

Program ini membandingkan dua metode pencarian data, yaitu sequential search dan binary search, dalam bahasa pemrograman Go. Data yang digunakan berbentuk slice bertipe float64, dan program mencari nilai target tertentu.

Sequential search dilakukan langsung tanpa perlu mengurutkan data, dengan cara memeriksa satu per satu elemen dari awal. Sedangkan binary search dilakukan setelah data diurutkan, dan mencari nilai dengan cara membagi dua bagian data secara berulang.

Program menampilkan hasil pencarian beserta jumlah langkah yang dibutuhkan oleh masing masing metode, sehingga pengguna bisa melihat perbedaan efisiensi antara pencarian linear dan pencarian biner.

```

package main

// 103112400024 Setyo Nugroho

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 {

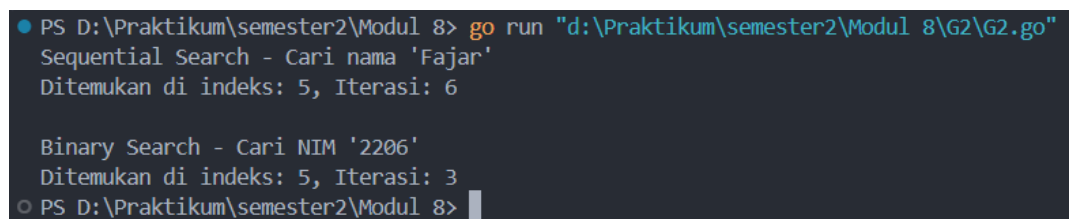
        fmt.Printf("Tidak ditemukan, Iterasi: %d\n", iterBin)

    }

}

```

### Output:



```

PS D:\Praktikum\semester2\Modul 8> go run "d:\Praktikum\semester2\Modul 8\G2\G2.go"
Sequential Search - Cari nama 'Fajar'
Ditemukan di indeks: 5, Iterasi: 6

Binary Search - Cari NIM '2206'
Ditemukan di indeks: 5, Iterasi: 3
PS D:\Praktikum\semester2\Modul 8>

```

### Deskripsi:

Program ini menggunakan bahasa Go untuk mencari data mahasiswa dengan dua metode: **Sequential Search** untuk pencarian berdasarkan nama, dan **Binary Search** untuk pencarian berdasarkan NIM. Data mahasiswa disimpan dalam array struct, lalu dicari menggunakan nama (tanpa perlu diurutkan) dan NIM (setelah data diurutkan). Hasil pencarian menunjukkan indeks data ditemukan dan jumlah iterasi, sehingga bisa dibandingkan efisiensi kedua metode.

## III. UNGUIDED

### 1

#### Source code:

```
package main
```

```
// 103112400024 Setyo Nugroho
```

```
import "fmt"

func pilkart() {
    var input int
    var totalMasuk, totalSah int
    var suara [21]int

    for {
        fmt.Scan(&input)
        if input == 0 {
            break
        }
        totalMasuk++

        if input >= 1 && input <= 20 {
            suara[input]++
            totalSah++
        }
    }

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

    for i := 1; i <= 20; i++ {
        if suara[i] > 0 {
```

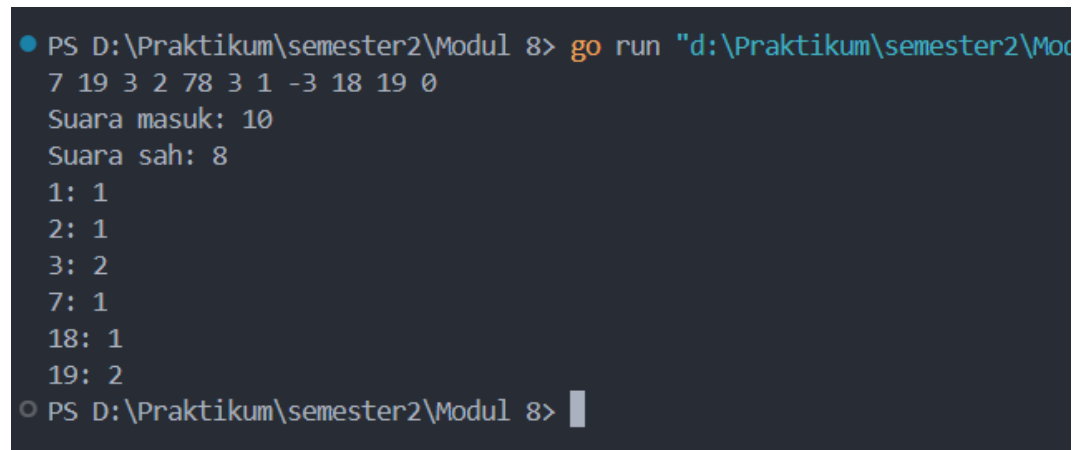
```

        fmt.Printf("%d: %d\n", i, suara[i])
    }
}

func main() {
    pilkart()
}

```

### Output:



```

PS D:\Praktikum\semester2\Modul 8> go run "d:\Praktikum\semester2\Modul 8\pilkart.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
PS D:\Praktikum\semester2\Modul 8>

```

### Deskripsi:

Program ini menerima input suara dari pengguna secara berulang. Setiap suara yang diberikan akan dihitung, dan suara yang sah adalah suara yang berada di antara angka 1 hingga 20. Program akan terus menerima input hingga pengguna memasukkan angka 0, yang menandakan akhir dari proses pemungutan suara.

2

**Source code:**

```
package main

// 103112400024 Setyo Nugroho

import "fmt"

func pilkart() {
    var input int
    var totalMasuk, totalSah int
    var suara [21]int

    for {
        fmt.Scan(&input)
        if input == 0 {
            break
        }
        totalMasuk++
        if input >= 1 && input <= 20 {
            suara[input]++
            totalSah++
        }
    }

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

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

```
ketua := 0
```

```
maxSuara := 0
```

```
for i := 1; i <= 20; i++ {
```

```
    if suara[i] > maxSuara || (suara[i] == maxSuara && i < ketua) {
```

```
        ketua = i
```

```
        maxSuara = suara[i]
```

```
    }
```

```
}
```

```
wakil := 0
```

```
secondMax := 0
```

```
for i := 1; i <= 20; i++ {
```

```
    if i != ketua {
```

```
        if suara[i] > secondMax || (suara[i] == secondMax && (wakil == 0  
|| i < wakil)) {
```

```
            wakil = i
```

```
            secondMax = suara[i]
```

```
        }
```

```
    }
```

```
}
```

```
fmt.Printf("Ketua RT: %d\n", ketua)
```

```
fmt.Printf("Wakil ketua: %d\n", wakil)
```



```
}
```

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

Output:

```
PS D:\Praktikum\semester2\Modul 8> go run "d:\Praktikum\semester2\Modul 8\UG2\UG2.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:\Praktikum\semester2\Modul 8>
```

Deskripsi:

Program ini menghitung hasil pemilihan ketua dan wakil ketua berdasarkan input suara. Suara yang valid adalah angka antara 1 hingga 20, dan program akan terus menerima input hingga angka 0 dimasukkan. Setelah itu, program akan menampilkan jumlah total suara masuk, suara sah, serta siapa yang terpilih sebagai ketua dan wakil ketua berdasarkan suara terbanyak. Jika ada suara sama, calon dengan nomor lebih kecil yang terpilih.

3

Source code:

```
package main  
  
// 103112400024 Setyo Nugroho  
  
import "fmt"
```

```
const NMAX = 1000000
```

```
var data [NMAX]int
```

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

```
func posisi(n, k int) int {  
    gold, exp := 0, n-1  
    for gold <= exp {  
        mid := (gold + exp) / 2  
        if data[mid] == k {  
            return mid  
        } else if data[mid] < k {  
            gold = mid + 1  
        } else {  
            exp = mid - 1  
        }  
    }  
    return -1  
}
```

```

func main() {
    var n, k int

    fmt.Scan(&n, &k)

    isiArray(n)

    result := posisi(n, k)

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

```

Output:

```

PS D:\Praktikum\semester2\Modul 8> go run "d:\Praktikum\semester2\Modul 8\UG3\UG3.go"
12 534
1 3 8 16 32 123 323 323 534 534 823 999
8
PS D:\Praktikum\semester2\Modul 8> go run "d:\Praktikum\semester2\Modul 8\UG3\UG3.go"
12 535
1 3 8 16 23 123 323 323 534 534 823 999
TIDAK ADA
PS D:\Praktikum\semester2\Modul 8>

```

Deskripsi:

Program ini mencari posisi angka k dalam array yang sudah terurut menggunakan algoritma **binary search**. Program pertama-tama menerima input jumlah elemen array n dan angka yang ingin dicari k. Setelah itu, program mengisi array dengan data yang diberikan oleh pengguna, lalu melakukan pencarian menggunakan binary search. Jika angka k ditemukan, program akan menampilkan indeksinya. Jika tidak, program akan menampilkan "TIDAK ADA".

#### IV. Kesimpulan

Di praktikum ini, kita belajar tentang dua cara untuk mencari data dalam sebuah array, yaitu **Sequential Search** dan **Binary Search**. Sequential Search bisa mencari elemen meskipun data dalam array belum diurutkan, tapi prosesnya lebih lama kalau data banyak. Sedangkan Binary Search lebih cepat karena hanya bisa digunakan pada array yang sudah terurut. Dengan Binary Search, pencarian bisa lebih efisien karena hanya memerlukan waktu lebih sedikit ( $O(\log n)$ ) dibandingkan Sequential Search yang memakan waktu lebih lama ( $O(n)$ ).

Praktikum ini tujuannya buat ngasih tahu cara kerja dua algoritma pencarian data, yaitu **Sequential Search** dan **Binary Search**. Di sini kita lihat gimana cara pengurutan data bisa mempengaruhi kecepatan pencarian, karena Binary Search cuma efektif kalau datanya udah terurut. Melalui praktikum ini, kita jadi paham gimana cara optimasi pencarian data dan memanfaatkan algoritma yang sesuai dengan kebutuhan.

#### V. REFERENSI

Modul 8