

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2
MODUL 11
PENCARIAN NILAI ACAK PADA HIMPUNAN DATA**



Oleh:

NAMA : HAKAN ISMAIL AFNAN

NIM : 103112400038

KELAS : 12-IF-01

**S1 TEKNIK INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO**

2025

I. DASAR TEORI

Metode Pencarian Data Acak dalam Bahasa Go

Pencarian data dalam himpunan nilai acak merupakan operasi penting dalam pemrograman, khususnya dalam bahasa Go yang menyediakan array dan slice sebagai struktur data utama. Metode pencarian yang umum digunakan adalah sekuensial search dan binary search.

Sekuensial Search memeriksa data satu per satu secara berurutan tanpa memerlukan data terurut. Metode ini mudah diimplementasikan dan sangat berguna untuk dataset kecil atau yang tidak terstruktur, namun kurang efisien untuk data besar karena waktu pencarian bertambah seiring jumlah data.

Binary Search memerlukan data yang sudah terurut dan menggunakan prinsip pembagian ruang pencarian menjadi dua bagian pada setiap langkah. Dengan cara ini, binary search mampu mengurangi jumlah perbandingan secara signifikan, sehingga sangat efisien untuk dataset besar.

Selain itu, dalam bahasa Go, array bertipe data struct memungkinkan penyimpanan data dengan banyak atribut. **Pencarian pada array struct** dilakukan dengan membandingkan atribut tertentu. Jika data terurut berdasarkan atribut tersebut, binary search dapat digunakan, jika tidak, sekuensial search menjadi pilihan utama.

Implementasi kedua metode ini dalam Go sangat membantu dalam pengelolaan data seperti pencarian nilai dalam daftar siswa, pencarian produk dalam katalog, dan lain-lain, sehingga meningkatkan efisiensi program secara signifikan.

II. GUIDED

Contoh 1

```
//Nama : Hakan Ismail Afnan
//NIM : 103112400038
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)
    }
}

```

Output:

```

PS C:\Users\User\Documents\HAKAN ISMAIL AFNAN\103112400038_MODULE8> go run "C:\Users\User\Documents\HAKAN ISMAIL AFNAN\103112400038_MODULE8\Guided8\1.go"
Sequential Search (data tidak perluurut)
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 C:\Users\User\Documents\HAKAN ISMAIL AFNAN\103112400038_MODULE8>

```

Penjelasan:

Program ini menggunakan bahasa Go untuk mencari nilai tertentu dalam array angka desimal dengan dua cara: pencarian sekuensial pada data acak dan pencarian binary pada data yang sudah diurutkan. Pada pencarian sekuensial, program memeriksa setiap elemen satu per satu hingga

menemukan nilai target atau habis data. Setelah data diurutkan, binary search mencari nilai target dengan membagi dua ruang pencarian secara berulang, sehingga lebih cepat.

Setiap langkah pencarian dicetak untuk memudahkan pemahaman proses. Program menampilkan hasil pencarian berupa indeks dan jumlah langkah yang dibutuhkan, memperlihatkan keunggulan binary search dalam efisiensi pencarian pada data terurut.

Contoh Soal 2

```
//Nama : Hakan Ismail Afnan
//NIM : 103112400038
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 C:\Users\User\Documents\HAKAN ISMAIL AFNAN\10311240038_MODULE8> go run "c:\Users\User\Documents\HAKAN ISMAIL AFNAN\10311240038_MODULE8\Guided0\2.go"
Sequential Search - Cari nama 'Fajar'
Ditemukan di indeks: 5, Iterasi: 6

Binary Search - Cari NIM '2206'
Ditemukan di indeks: 5, Iterasi: 3
PS C:\Users\User\Documents\HAKAN ISMAIL AFNAN\10311240038_MODULE8>

```

Penjelasan

Program ini menggunakan bahasa Go untuk mencari data mahasiswa dalam array bertipe struct. Pencarian pertama menggunakan sequential search yang mencari nama mahasiswa secara berurutan. Pencarian kedua menggunakan binary search yang mencari NIM mahasiswa setelah data diurutkan berdasarkan NIM.

Metode sequential search cocok untuk data yang tidak terurut, sedangkan binary search lebih cepat namun membutuhkan data terurut. Program menampilkan indeks hasil pencarian dan jumlah langkah yang diperlukan, membantu memahami perbedaan efisiensi kedua metode.

III. UNGUIDED

Soal 1

```
// Nama   : Hakan Ismail Afnan
// NIM    : 103112400038
package main

import (
    "fmt"
    "sort"
)

func binarySearch(arr []int, target int) bool {
    low, high := 0, len(arr)-1
    for low <= high {
        mid := (low + high) / 2
        if arr[mid] == target {
            return true
        } else if arr[mid] < target {
            low = mid + 1
        } else {
            high = mid - 1
        }
    }
    return false
}

func main() {
    const maxCalon = 20
    data := []int{7, 19, 3, 2, 78, 3, 1, 3, 18, 19, 0} // data
    input langsung

    // Daftar calon sudah terurut 1..20
    validCalon := make([]int, maxCalon)
    for i := 1; i <= maxCalon; i++ {
        validCalon[i-1] = i
    }
    sort.Ints(validCalon) // sebenarnya sudah terurut, tapi
    untuk konsistensi

    countSuara := make(map[int]int)
    totalData := 0
    totalValid := 0

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

```

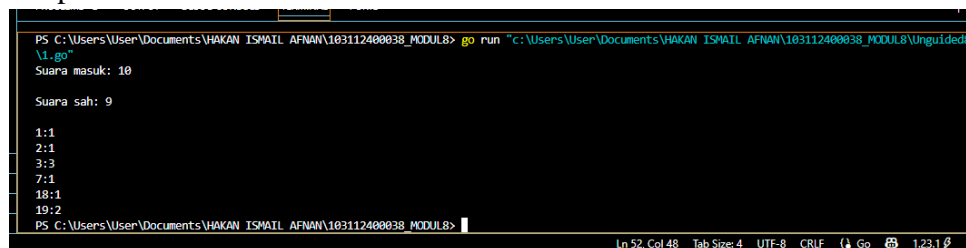
        if v == 0 {
            break
        }
        totalData++
        if binarySearch(validCalon, v) {
            countSuara[v]++
            totalValid++
        }
    }

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

    for i := 1; i <= maxCalon; i++ {
        if count, ok := countSuara[i]; ok {
            fmt.Printf("%d:%d\n", i, count)
        }
    }
}

```

Output:



```

PS C:\Users\User\Documents\HAKAN ISMAIL AFNAN\103112400038_MODUL8> go run "c:\Users\User\Documents\HAKAN ISMAIL AFNAN\103112400038_MODUL8\Unguided8\1.go"
Suara masuk: 10
Suara sah: 9
1:1
2:1
3:3
7:1
18:1
19:2
PS C:\Users\User\Documents\HAKAN ISMAIL AFNAN\103112400038_MODUL8>

```

Penjelasan

Program ini bertujuan menghitung dan memvalidasi suara pemilihan ketua RT dari data input berupa nomor calon. Data suara yang masuk berupa deretan angka, di mana angka 0 menandai akhir data. Program menggunakan metode binary search untuk memvalidasi apakah nomor suara termasuk dalam daftar calon yang sah (nomor 1 sampai 20). Pertama, program membuat daftar calon yang sudah terurut dari 1 hingga 20. Kemudian, untuk setiap suara yang diterima sebelum angka 0, program melakukan pencarian menggunakan binary search pada daftar calon tersebut. Jika suara valid, maka suara tersebut dihitung dan disimpan dalam struktur data map untuk menghitung jumlah suara tiap calon. Setelah seluruh data diproses, program menampilkan jumlah total suara yang masuk, jumlah suara sah, dan rincian jumlah suara yang diperoleh masing-masing calon.

Soal 2

```
//Nama : Hakan Ismail Afnan
//NIM : 103112400038
package main

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

func binarySearch(arr []int, target int) bool {
    low, high := 0, len(arr)-1
    for low <= high {
        mid := (low + high) / 2
        if arr[mid] == target {
            return true
        } else if arr[mid] < target {
            low = mid + 1
        } else {
            high = mid - 1
        }
    }
    return false
}

func main() {
    const maxCalon = 20
    validCalon := make([]int, maxCalon)
    for i := 1; i <= maxCalon; i++ {
        validCalon[i-1] = i
    }
    sort.Ints(validCalon)

    countSuara := make(map[int]int)
    totalData, totalValid := 0, 0

    fmt.Println("Masukkan suara (akhiri dengan 0):")
    scanner := bufio.NewScanner(os.Stdin)
    if scanner.Scan() {
        input := scanner.Text()
        data := strings.Fields(input)
```

```

        for _, v := range data {
            num, err := strconv.Atoi(v)
            if err != nil {
                continue
            }
            if num == 0 {
                break
            }
            totalData++
            if binarySearch(validCalon, num) {
                countSuara[num]++
                totalValid++
            }
        }
    }
}

type calon struct {
    no, suara int
}

var calonList []calon
for no, suara := range countSuara {
    calonList = append(calonList, calon{no, suara})
}

sort.Slice(calonList, func(i, j int) bool {
    if calonList[i].suara == calonList[j].suara {
        return calonList[i].no < calonList[j].no
    }
    return calonList[i].suara > calonList[j].suara
}))

ketua := calonList[0].no
wakil := calonList[1].no

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

```

Output:

```

PS C:\Users\User\Documents\HAKAN ISMAIL AFNAN\103112400038_MODUL8> go run "c:\Users\User\Documents\HAKAN ISMAIL AFNAN\103112400038_MODUL8\Unguided\2.go"
Masukkan suara (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:\Users\User\Documents\HAKAN ISMAIL AFNAN\103112400038_MODUL8>

```

Penjelasan

Program ini menggunakan metode binary search untuk memvalidasi suara yang masuk sebelum menghitung jumlah suara tiap calon. Input suara dibaca dalam satu baris dan setiap suara diperiksa keabsahannya dengan binary search pada daftar calon yang sudah terurut (nomor 1 sampai 20).

Setelah validasi, suara yang sah dihitung dan disimpan dalam struktur data map. Selanjutnya, program mengurutkan data calon berdasarkan jumlah suara secara menurun dan nomor calon secara menaik jika terjadi seri, sehingga pemenang ketua dan wakil ketua dapat ditentukan dengan mudah.

Soal 3

```
//Nama : Hakan Ismail Afnan
//NIM : 103112400038
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 {
    for i := 0; i < n; i++ {
        if data[i] == k {
            return i
        }
    }
    return -1
}
```

Output:

```
PS C:\Users\User\Documents\HAKAN ISMAIL AFNAN\103112400038_MODULE8> go run "c:\Users\User\Documents\HAKAN ISMAIL AFNAN\103112400038_MODULE8\Unguided8\3.go"
12 534
1 3 8 16 32 123 323 323 534 543 823 999
8
PS C:\Users\User\Documents\HAKAN ISMAIL AFNAN\103112400038_MODULE8> go run "c:\Users\User\Documents\HAKAN ISMAIL AFNAN\103112400038_MODULE8\Unguided8\3.go"
12 535
1 3 8 16 32 123 323 323 534 543 823 999
TIDAK ADA
PS C:\Users\User\Documents\HAKAN ISMAIL AFNAN\103112400038_MODULE8>
```

Penjelasan:

Program ini menggunakan metode pencarian sekuensial, yaitu dengan memeriksa setiap elemen dalam array satu per satu secara berurutan dari indeks 0 hingga indeks $n-1$. Program membaca input berupa banyaknya data n , nilai yang dicari k , dan data array yang sudah terurut membesar.

Fungsi `isiArray` bertugas mengisi array dengan data yang dibaca dari input. Fungsi `posisi` melakukan pencarian linear dengan membandingkan setiap elemen array dengan nilai k . Jika ditemukan, fungsi mengembalikan indeks posisi elemen tersebut; jika tidak ditemukan, mengembalikan -1 .

IV. KESIMPULAN

Pencarian sekuensial dan binary search adalah dua pendekatan utama dalam pencarian data pada himpunan nilai acak. Sequential search mudah diterapkan tanpa syarat data terurut, tetapi kurang efisien untuk data besar karena harus memeriksa setiap elemen. Binary search, meskipun memerlukan data terurut, menawarkan pencarian yang jauh lebih cepat dengan membagi ruang pencarian secara sistematis. Dengan demikian, binary search lebih cocok untuk data terurut dan berukuran besar, sementara sekuensial search lebih praktis untuk data kecil atau tidak terurut. Kedua metode ini saling melengkapi dan penggunaannya harus disesuaikan dengan kondisi data dan kebutuhan efisiensi.

REFERENSI

Modul 11 Algoritma Pemrograman Pencarian Nilai Acak pada Himpunan Data