

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



Oleh:

NAMA: NUFAIL ALAUDDIN TSAQIF

NIM: 103112400084

KELAS: IF-12-01

S1 TEKNIK INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

I. DASAR TEORI

1. Sequential Search

Sequential Search adalah metode pencarian yang sederhana, di mana elemen-elemen dalam array diperiksa satu per satu hingga data yang dicari ditemukan atau seluruh array telah diperiksa. Pencarian dimulai dari elemen pertama dan berlanjut hingga akhir array. Kelebihan dari algoritma ini adalah dapat diterapkan pada array yang tidak terurut, namun memiliki kelemahan dalam efisiensi dengan kompleksitas waktu $O(n)$, yang menyebabkan pencarian menjadi lebih lama seiring bertambahnya jumlah elemen dalam array. Oleh karena itu, meskipun mudah diterapkan, algoritma ini kurang efisien untuk dataset besar.

2. Binary Search

Binary Search adalah algoritma pencarian yang lebih efisien daripada Sequential Search, tetapi hanya dapat diterapkan pada array yang sudah terurut. Algoritma ini bekerja dengan membagi array terurut menjadi dua bagian, kemudian memilih bagian yang relevan berdasarkan perbandingan dengan nilai tengah. Jika nilai tengah lebih kecil dari nilai yang dicari, pencarian dilanjutkan ke setengah kanan, jika lebih besar, ke setengah kiri. Proses ini diulang hingga data ditemukan atau rentang pencarian habis. Keunggulan Binary Search terletak pada kompleksitas waktunya yang lebih rendah, yaitu $O(\log n)$, sehingga lebih efisien untuk pencarian dalam dataset besar.

3. Pencarian pada Data Bertipe Struct

Pencarian pada array bertipe data struct pada dasarnya mirip dengan tipe data dasar, dengan tambahan kategori yang akan dicari. Pada Binary Search, array harus terurut sesuai dengan kategori pencarian yang dilakukan. Misalnya, jika pencarian dilakukan berdasarkan NIM mahasiswa, maka array harus diurutkan berdasarkan NIM. Binary Search tidak akan bekerja dengan benar jika array terurut berdasarkan NIM tetapi pencarian dilakukan berdasarkan nama atau kategori lainnya. Solusinya adalah mengurutkan array sesuai kategori yang dicari atau menggunakan Sequential Search jika diperlukan.

GUIDED 1

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
```

```

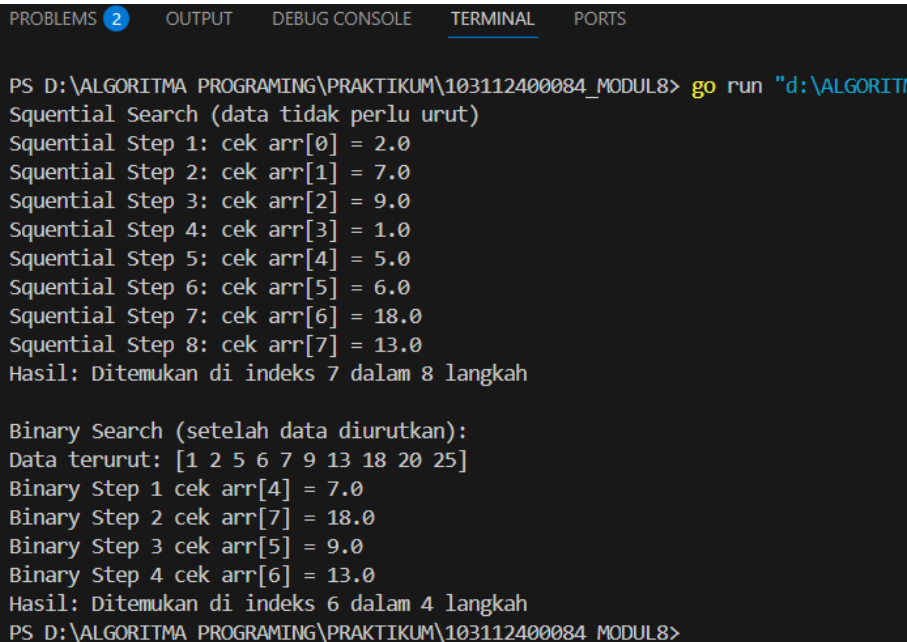
    fmt.Println("Sequential Search (data tidak perluurut)")
    idxSeq, iterSeq := sequentialSearch(data, target)
    if idxSeq != 1 {
        fmt.Printf("Hasil: Ditemukan di indeks %d dalam %d langkah\n\n", idxSeq,
iterSeq)
    } else {
        fmt.Printf("Hasil: Tidak ditemukan setelah %d langkah\n\n", 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)
    }
}

```

OUTPUT:



```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\ALGORITMA PROGRAMING\PRAKTIKUM\103112400084_MODUL8> go run "d:\ALGORITMA
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 D:\ALGORITMA PROGRAMING\PRAKTIKUM\103112400084_MODUL8>

```

DEKSRIPSI:

Program ini melakukan pencarian elemen dalam array menggunakan dua metode: Sequential Search dan Binary Search. Pada Sequential Search, program memeriksa setiap elemen satu per satu tanpa memerlukan data yang terurut, sambil mencatat jumlah langkah pencarian. Setelah itu, array diurutkan, dan Binary Search diterapkan pada array yang terurut. Pencarian biner lebih efisien karena membagi array menjadi dua bagian pada setiap langkah dan membatasi pencarian ke setengah bagian yang relevan. Program menampilkan langkah-langkah pencarian serta hasil akhir, termasuk jumlah langkah yang dibutuhkan atau menyatakan bahwa elemen tidak ditemukan.

GUIDED 2

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)
    }

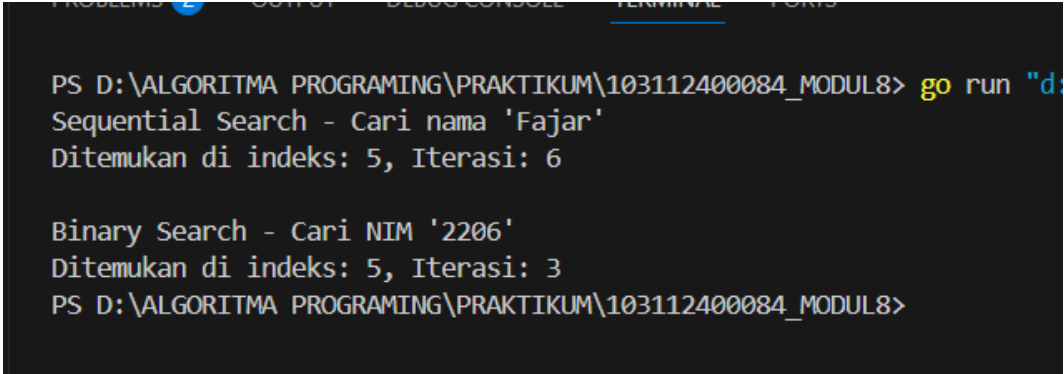
    // Urutkan data berdasarkan NIM untuk binary search
    sort.Slice(data[:n], func(i, j int) bool {
        return data[i].nim < data[j].nim
    })
}

```

```
// 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)
}
}
```

OUTPUT:



```
PS D:\ALGORITMA PROGRAMING\PRAKTIKUM\103112400084_MODUL8> go run "d:
Sequential Search - Cari nama 'Fajar'
Ditemukan di indeks: 5, Iterasi: 6

Binary Search - Cari NIM '2206'
Ditemukan di indeks: 5, Iterasi: 3
PS D:\ALGORITMA PROGRAMING\PRAKTIKUM\103112400084_MODUL8>
```

DEKSRIPSI:

Program ini mencari data mahasiswa (nama, NIM, kelas, jurusan, dan IPK) menggunakan Sequential Search dan Binary Search. Pada Sequential Search, program memeriksa elemen secara berurutan berdasarkan nama, mencatat jumlah iterasi. Sedangkan pada Binary Search, setelah data diurutkan, pencarian dilakukan dengan membagi data secara efisien untuk menemukan NIM. Program ini menampilkan hasil pencarian beserta jumlah langkah yang dibutuhkan dalam masing-masing metode.

4. UNGUIDED

UNGUIDED 1

CODE

```
package main
// 103112400084
// NUFAIL ALAUDDIN TSAQIF
import "fmt"

func pilkart() {
    var target int
    var suaraMasuk, suaraSah int
    jumlahVote := make([]int, 20)

    for {
        _, err := fmt.Scan(&target)
        if err != nil || target < 0 {
            continue
        }
        if target == 0 {
            break
        }
        suaraMasuk++
        if target >= 1 && target <= 20 {
            suaraSah++
            jumlahVote[target-1]++
        }
    }

    fmt.Printf("\nSuara masuk: %d\n", suaraMasuk)
    fmt.Printf("Suara sah: %d\n", suaraSah)

    fmt.Println("\nHasil Perhitungan Suara:")
    for i, suara := range jumlahVote {
        if suara > 0 {
            fmt.Printf("Calon %d: %d suara\n", i+1, suara)
        }
    }
}

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

OUTPUT

```
PROBLEMS 6 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\ALGORITMA PROGRAMING\PRAKTIKUM\103112400084_MODUL8> go run "c
7 19 3 2 78 3 1 -3 18 19 0

Suara masuk: 9
Suara sah: 8

Hasil Perhitungan Suara:
Calon 1: 1 suara
Calon 2: 1 suara
Calon 3: 2 suara
Calon 7: 1 suara
Calon 18: 1 suara
Calon 19: 2 suara
PS D:\ALGORITMA PROGRAMING\PRAKTIKUM\103112400084_MODUL8> |
```

DEKSRIPSI

Program ini mensimulasikan pemungutan suara untuk memilih ketua dalam sebuah pemilu, di mana setiap calon diidentifikasi dengan angka. Program akan terus menerima input suara hingga angka 0 dimasukkan, menandakan akhir pemilihan. Setiap suara dihitung sebagai suara masuk, dan jika suara tersebut valid (antara 1 hingga 20), dihitung sebagai suara sah dan dikaitkan dengan calon yang dipilih. Program menggunakan tipe data map untuk menyimpan jumlah suara yang diterima oleh masing-masing calon, dan setelah pemilihan selesai, program mencetak jumlah total suara yang sah, tidak sah, serta hasil perhitungan suara yang diterima oleh setiap calon.

UNGUIDED 2

CODE

```
package main
// 103112400084
// NUFAIL ALAUDDIN TSAQIF
import "fmt"

func pilkart() {
    var target, suaraMasuk, suaraSah int
    jumlahVote := make(map[int]int)

    for {
        _, err := fmt.Scan(&target)
        if err != nil || target == 0 {
            break
        }
        if target >= 1 && target <= 20 {
            suaraMasuk++
            suaraSah++
            jumlahVote[target]++
        } else {
            suaraMasuk++
        }
    }

    fmt.Println("Suara masuk:", suaraMasuk)
    fmt.Println("Suara sah:", suaraSah)

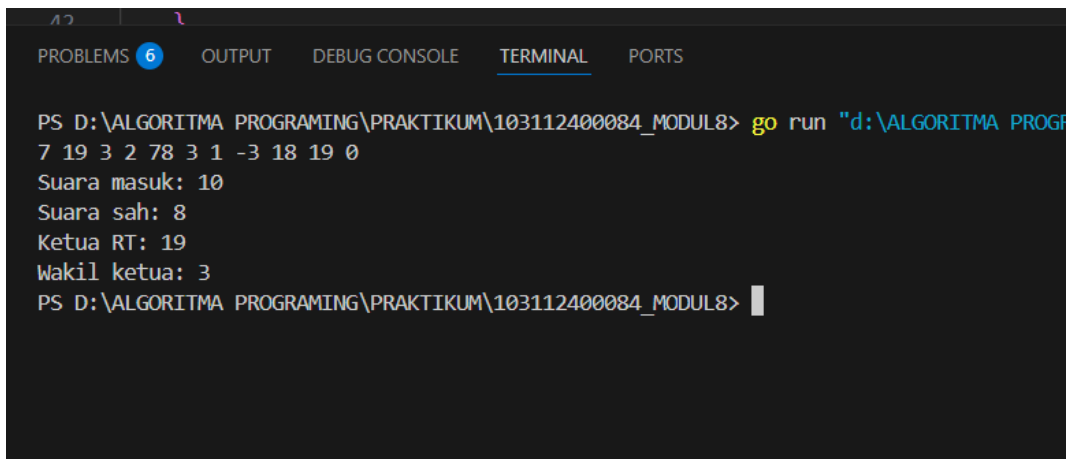
    var ketua, wakilKetua int
    maxVote, secondMaxVote := 0, 0

    for calon, vote := range jumlahVote {
        if vote > maxVote {
            secondMaxVote = maxVote
            wakilKetua = ketua
            maxVote = vote
            ketua = calon
        } else if vote > secondMaxVote {
            secondMaxVote = vote
            wakilKetua = calon
        }
    }
}
```

```
    fmt.Println("Ketua RT:", ketua)
    fmt.Println("Wakil ketua:", wakilKetua)
}

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

OUTPUT

A screenshot of a terminal window showing the execution of a Go program. The terminal has tabs for PROBLEMS (6), OUTPUT, DEBUG CONSOLE, TERMINAL (selected), and PORTS. The command executed is 'go run "d:\ALGORITMA PROGRAMING\PRAKTIKUM\103112400084_MODUL8>'. The output shows a list of numbers: '7 19 3 2 78 3 1 -3 18 19 0', followed by 'Suara masuk: 10', 'Suara sah: 8', 'Ketua RT: 19', and 'Wakil ketua: 3'. The prompt 'PS D:\ALGORITMA PROGRAMING\PRAKTIKUM\103112400084_MODUL8>' is visible at the bottom.

```
PS D:\ALGORITMA PROGRAMING\PRAKTIKUM\103112400084_MODUL8> go run "d:\ALGORITMA PROGRAMING\PRAKTIKUM\103112400084_MODUL8>
7 19 3 2 78 3 1 -3 18 19 0
Suara masuk: 10
Suara sah: 8
Ketua RT: 19
Wakil ketua: 3
PS D:\ALGORITMA PROGRAMING\PRAKTIKUM\103112400084_MODUL8>
```

DEKSRIPSI

Program ini mensimulasikan pemilihan ketua dan wakil ketua untuk suatu organisasi, seperti RT, dengan menghitung suara yang diberikan oleh peserta. Suara yang valid (antara 1 dan 20) dihitung sebagai suara sah, sementara suara di luar rentang tersebut dianggap tidak sah. Setelah semua suara dihitung, program menentukan ketua dan wakil ketua berdasarkan jumlah suara terbanyak, di mana ketua adalah calon dengan suara terbanyak dan wakil ketua adalah calon dengan suara terbanyak kedua. Program juga menampilkan jumlah suara yang masuk, suara sah, serta hasil pemilihan ketua dan wakil ketua.

UNGUIDED 3

CODE

```
package main
// 103112400084
// NUFAIL ALAUDDIN TSAQIF
import "fmt"

const NMAX = 10000

var data [NMAX]int

func main() {
    var n, k int
    fmt.Scan(&n, &k)
    isiArray(n)

    letak := posisi(n, k)
    if letak == -1 {
        fmt.Print("TIDAK ADA")
    } else {
        fmt.Print(letak)
    }
}

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

func posisi(a, b int) int {
    low := 0
    high := a - 1
    for low <= high {
        mid := (low + high) / 2
        if data[mid] == b {
            return mid
        } else if b < data[mid] {
            high = mid - 1
        } else {
            low = mid + 1
        }
    }
    return -1
}
```

```
}
```

OUTPUT

```
PS D:\ALGORITMA PROGRAMING\PRAKTIKUM\103112400084_MODUL8> go run "d:\ALGORITMA
12 534
1 3 8 16 32 123 323 323 534 543 823 999
8
PS D:\ALGORITMA PROGRAMING\PRAKTIKUM\103112400084_MODUL8> go run "d:\ALGORITMA
12 535
1 3 8 16 32 123 323 323 534 543 823 999
TIDAK ADA
PS D:\ALGORITMA PROGRAMING\PRAKTIKUM\103112400084_MODUL8> |
```

DEKSRIPSI

Program ini mencari angka dalam array dengan dua langkah utama: penyortiran dan pencarian biner. Pertama, array diurutkan menggunakan Bubble Sort, kemudian pencarian dilakukan dengan Binary Search yang membagi array secara efisien. Jika angka ditemukan, program mencetak indeksinya; jika tidak, menampilkan pesan "TIDAK ADA". Program ini menggabungkan teknik penyortiran dan pencarian efisien pada data terurut.

5. KESIMPULAN

Pada praktikum ini, telah dipelajari dan diterapkan dua algoritma pencarian data, yaitu Sequential Search dan Binary Search menggunakan bahasa pemrograman Go. Sequential Search memeriksa elemen secara berurutan hingga data ditemukan, sementara Binary Search diterapkan pada data terurut dengan membandingkan elemen tengah untuk mempersempit ruang pencarian. Pemahaman kedua algoritma ini penting dalam pengolahan data untuk efisiensi pencarian, serta memberikan wawasan mengenai perbedaan efisiensi antara pencarian sederhana dan pencarian optimal pada struktur data terurut.

REFERENSI

MODUL 11. PENCARIAN NILAI ACAK PADA HIMPUNAN
DATA