LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL 8

PENCARIAN NILAI ACAK PADA HIMPUNAN DATA

Oleh:

Muhammad Faris Rachmadi

103112400079

IF12-01

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

I. DASAR TEORI

8.1 Sequential Search

Pencarian secara sekuensial ini adalah pencarian yang dilakukan dari data pertama, kedua hingga terakhir secara satu persatu dan berurutan. Ciri khas dari pencarian ini adalah proses pencarian akan berhenti ketika data yang dicari ditemukan, walaupun masih terdapat data yang belum dicek nilainya. Algoritma ini dikenal dengan nama Sequential Search, karena prosesnya melakukan pengecekan setiap elemen array secara satu persatu dan sekuensial dari data pertama hingga ditemukan atau data terakhir.

1) Asumsi terdapat suatu array of integer T dengan indeks dari 0 hingga N-1, dan suatu nilai yang dicari pada array T, yaitu X.

2) Status pencarian digunakan untuk menandakan data yang dicari ditemukan atau tidak, misalnya found dengan tipe boolean.

3) Pencarian dilakukan dari T[0] sampai ke T[N-1], setiap kali perbandingan dengan X, update nilai found.

4) Perulangan harus dihentikan apabila status pencarian found bernilai true (data ditemukan) atau T[N-1] telah dicek.

8.2 Binary Search

Ide algoritma adalah: (dengan asumsi data terurut dari kecil membesar (ascending), dan data dengan indeks kecil ada di "kiri" dan indeks besar ada di "kanan")

1) Ambil salah satu data dalam rentang data yang ada, algoritma di bawah menggunakan rentang dari kiri ***kr*** s.d. kanan ***kn***. Untuk kemudahan dan alasan lainnya, biasanya yang diambil adalah paling tengah dalam rentang tersebut.

2) Jika data terambil tersebut terlalu kecil, maka ubah/geser rentang data ke sebelah kanan posisi data tersebut. Ini karena jika data terambil terlalu kecil, maka semua data sebalah kirinya juga akan terlalu kecil dari yang ingin dicari.

3) Begitu juga sebaliknya jika data terambil terlalu besar.

8.3 Pencarian pada Array Bertipe Data Struct

Algoritma pencarian pada array bertipe data struct tidak berbeda jauh dengan tipe data dasar, kita hanya perlu menambahkan field kategori dari pencarian yang akan dilakukan. Khusus untuk algoritma binary search, maka keterurutan array harus sama dengan field kategori dari pencariannya. Misalnya apabila pencarian berdasarkan nim mahasiswa, maka algoritma binary search hanya akan bisa digunakan apabila array terurut berdasarkan nim. Algoritma binary search tidak akan berjalan apabila array terurut berdasarkan nim mahasiswa, tetapi pencarian yang dilakukan berdasarkan filed nama atau kategori yang lain selain nim. Solusinya adalah array harus diurutkan berdasarkan nim terlebih dahulu, atau gunakan algoritma sequential search.

II. GUIDED

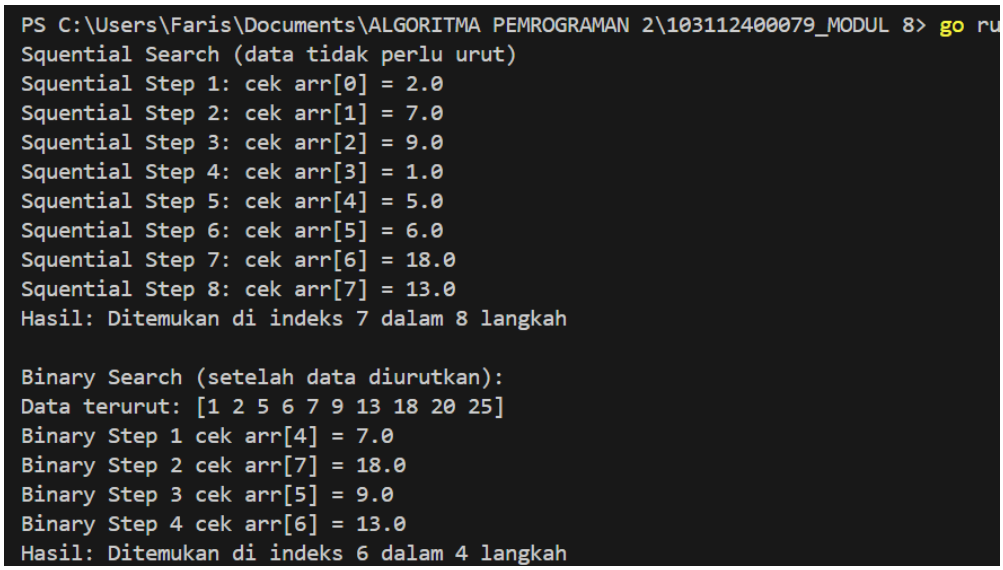
Guided 1

Code:

```
package main

import (
    "fmt"
    "sort"
)

func sequentialSearch(arr []float64, target float64) (int, int) {
    iterations := 0
    for i, val := range arr {
        iterations++
        fmt.Printf("Squential 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:



Deskripsi:

Program Go di atas mengimplementasikan dua metode pencarian: *sequential search* dan *binary search*, masing-masing untuk mencari nilai dalam sebuah array bertipe float64. Pada fungsi sequentialSearch, pencarian dilakukan dengan memeriksa setiap elemen dalam array satu per satu, dengan mencatat langkah yang dilakukan pada setiap iterasi. Hasilnya adalah indeks elemen yang ditemukan atau -1 jika tidak ditemukan, beserta jumlah iterasi yang dilakukan. Fungsi binarySearch hanya dapat digunakan setelah array diurutkan, dan bekerja dengan membagi array secara berulang menjadi dua bagian, memeriksa nilai tengah, dan mempersempit pencarian berdasarkan hasil perbandingan. Program ini mencetak hasil pencarian dari kedua metode beserta jumlah langkah yang dibutuhkan. Setelah array diurutkan menggunakan sort.Float64s, kedua metode tersebut diterapkan, dengan hasil akhir menunjukkan apakah elemen target ditemukan atau tidak.

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 C:\Users\Faris\Documents\ALGORITMA PEMROGRAMAN 2\103112400079_MODU
Sequential Search - Cari nama 'Fajar'
Ditemukan di indeks: 5, Iterasi: 6

Binary Search - Cari NIM '2206'
Ditemukan di indeks: 5, Iterasi: 3
```

Deskripsi:

Program Go di atas mengimplementasikan dua metode pencarian: *sequential search* dan *binary search*, yang digunakan untuk mencari data mahasiswa dalam array berdasarkan dua atribut yang berbeda, yaitu nama dan NIM. Pada fungsi SeqSearch_3, pencarian dilakukan dengan membandingkan nama mahasiswa satu per satu, dan mencatat langkah atau iterasi yang dilakukan hingga ditemukan atau tidak ditemukan. Fungsi BinarySearch_3 digunakan untuk mencari mahasiswa berdasarkan NIM setelah array diurutkan terlebih dahulu. Untuk mengurutkan data berdasarkan NIM, digunakan fungsi sort.Slice yang mengatur data berdasarkan urutan NIM mahasiswa. Program ini kemudian mencetak hasil pencarian untuk kedua metode tersebut, yaitu nama yang dicari dengan metode sequential search dan NIM yang dicari dengan metode binary search, beserta jumlah iterasi yang dilakukan. Output dari program menunjukkan apakah data ditemukan dan di indeks mana, atau jika tidak ditemukan, berapa banyak langkah pencarian yang dilakukan.

III. UNGUIDED

Unguided 1

Code:

```
package main

// Muhammad Faris Rachmadi
// 103112400079
import "fmt"

func pilkart() {
   var target int
   var suaraMasuk, suaraSah int = 0, 0
   var jumlahVote [21]int // Array untuk menyimpan suara calon 1-20

   for {
      fmt.Scan(&target)
      if target == 0 {
         break
      }
      if target >= 1 && target <= 20 {
         suaraSah++
         jumlahVote[target]++
      }
      suaraMasuk++
   }

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

   for i := 1; i <= 20; i++ {
      if jumlahVote[i] > 0 {
         fmt.Printf("%d: %d\n", i, jumlahVote[i])
      }
   }
}

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

Output:

```
PS C:\Users\Faris\Documents\ALGORITMA PEMROGRAMAN 2\103112400079_MODUL 8>
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 Go ini mensimulasikan pemilu dengan menghitung suara yang diterima oleh calon dari input angka 1 hingga 20. Fungsi pilkart menerima input suara, menghitung suara sah, dan mencatat jumlah suara untuk setiap calon dalam array. Program berhenti saat input 0 diterima. Di akhir, program menampilkan jumlah suara yang masuk, suara sah, dan suara yang diterima oleh setiap calon yang mendapat suara. Fungsi main hanya memanggil pilkart untuk menjalankan proses tersebut.

Unguided 2

Code:

```
package main

// Muhammad Faris Rachmadi
// 103112400079

import "fmt"

func pilkart() {
    var target int
    var suaraMasuk, suaraSah int
    var jumlahVote [21]int

    fmt.Println("Masukkan suara (akhiri dengan angka 0):")
    for {
        fmt.Scan(&target)
        if target == 0 {
            break
        }
        suaraMasuk++

        if target >= 1 && target <= 20 {
            suaraSah++
            jumlahVote[target]++
        }
    }

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

    var max1, max2 int
    var calonKetua, calonWakil int

    for i := 1; i <= 20; i++ {
        if jumlahVote[i] > max1 {
            max2 = max1
            calonWakil = calonKetua
            max1 = jumlahVote[i]
            calonKetua = i
        } else if jumlahVote[i] > max2 {
            max2 = jumlahVote[i]
            calonWakil = i
        }
```

```
    }

    fmt.Printf("Ketua RT: %d\n", calonKetua)
    fmt.Printf("Wakil ketua: %d\n", calonWakil)
}

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

Output:

```
PS C:\Users\Faris\Documents\ALGORITMA PEMROGRAMAN 2\103112400079_MODUL 8> g
Masukkan suara (akhiri dengan angka 0):
7 19 3 2 78 3 1 -3 18 19 0
Suara masuk: 10
Suara sah: 8
Ketua RT: 3
Wakil ketua: 19
```

Deskripsi:

Program Go ini adalah simulasi pemilihan ketua dan wakil ketua RT. Fungsi pilkart menerima input suara dalam bentuk angka dari 1 hingga 20 yang mewakili calon, dengan suara 0 menandakan akhir dari pemungutan suara. Program menghitung jumlah suara yang masuk dan suara sah. Setelah pemungutan suara selesai, program menentukan calon ketua dan wakil ketua berdasarkan jumlah suara terbanyak. Proses ini dilakukan dengan mencari dua calon dengan suara terbanyak menggunakan dua variabel max1 dan max2 untuk menyimpan jumlah suara terbanyak dan kedua terbanyak. Fungsi main hanya memanggil fungsi pilkart untuk menjalankan proses tersebut. Hasil akhirnya adalah menampilkan calon ketua dan wakil ketua beserta jumlah suara yang mereka peroleh.

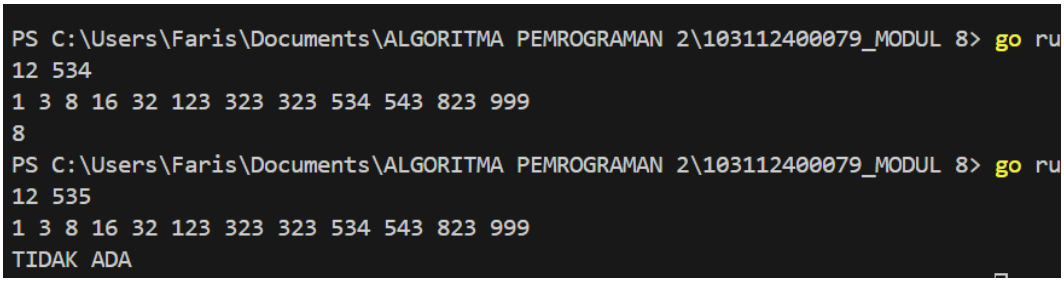
Unguided 3

Code:

```
package main

// Muhammad Faris Rachmadi
// 103112400079

import "fmt"

const NMAX = 1000000

var data [NMAX]int

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

   isiArray(n)
   hasil := posisi(n, k)

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

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

func posisi(n, k int) int {
   kiri, kanan := 0, n-1

   for kiri <= kanan {
      tengah := (kiri + kanan) / 2
      if data[tengah] == k {
         return tengah
      } else if data[tengah] < k {
         kiri = tengah + 1
      } else {
         kanan = tengah - 1
      }
```

```
    }

    return -1
}
```

Output:



Deskripsi:

Program Go ini mengimplementasikan algoritma pencarian biner (*binary search*) untuk menemukan posisi sebuah angka dalam array yang telah terurut. Program menerima dua input, yaitu jumlah elemen array n dan nilai yang ingin dicari k. Kemudian, array data diisi dengan nilai-nilai yang dimasukkan oleh pengguna. Fungsi posisi menggunakan metode binary search dengan membagi array menjadi dua bagian pada setiap iterasi, mempersempit pencarian berdasarkan perbandingan nilai tengah. Jika nilai k ditemukan, program mengembalikan indeksnya; jika tidak, program mencetak "TIDAK ADA".

IV. KESIMPULAN

Laporan praktikum ini membahas algoritma pencarian nilai acak pada himpunan data dengan menggunakan dua metode, yaitu *sequential search* dan *binary search*. Tujuan dari pencarian ini adalah untuk menemukan posisi atau keberadaan suatu nilai dalam array atau himpunan data. Pada metode *sequential search*, pencarian dilakukan dengan memeriksa setiap elemen array satu per satu hingga nilai yang dicari ditemukan. Sementara itu, *binary search* lebih efisien karena hanya dapat diterapkan pada data yang terurut, di mana pencarian dilakukan dengan membagi array menjadi dua bagian dan secara bertahap mengurangi ruang pencarian. Praktikum ini juga mencakup penerapan pencarian pada array bertipe data struct, seperti pencarian berdasarkan nama atau NIM mahasiswa, serta simulasi pemilu untuk menghitung suara sah menggunakan algoritma yang tepat.

V. REFRENSI

MODUL 11 PRAKTIKUM ALGORITMA PEMROGRAMAN

2 – PENCARIAN NILAI ACAK PADA HIMPUNAN DATA