LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL 8

PENCARIAN NILAI ACAK PADA HIMPUNAN DATA

Oleh:

HISYAM NURDIATMOKO

103112400049

IF-12-01

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

I. DASAR TEORI

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.

- Asumsi terdapat suatu array of integer T dengan indeks dari 0 hingga N-1, dan suatu nilai yang dicari pada array T, yaitu X.
- Status pencarian digunakan untuk menandakan data yang dicari ditemukan atau tidak, misalnya found dengan tipe boolean.
- Pencarian dilakukan dari T[0] sampai ke T[N-1], setiap kali perbandingan dengan X, update nilai found.
- Perulangan harus dihentikan apabila status pencarian found bernilai true (data ditemukan) T[N-1] telah dicek.

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")

- Ambil salah satu data dalam rentang data yang ada, algoritma di bawah menggunakan rentang dari kiri s.d. kanan. Untuk kemudahan dan alasan lainnya, biasanya yang diambil adalah paling tengah dalam rentang tersebut.
- 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.
- Begitu juga sebaliknya jika data terambil terlalu besar.

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

Source code Guided 1:

```
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}
```

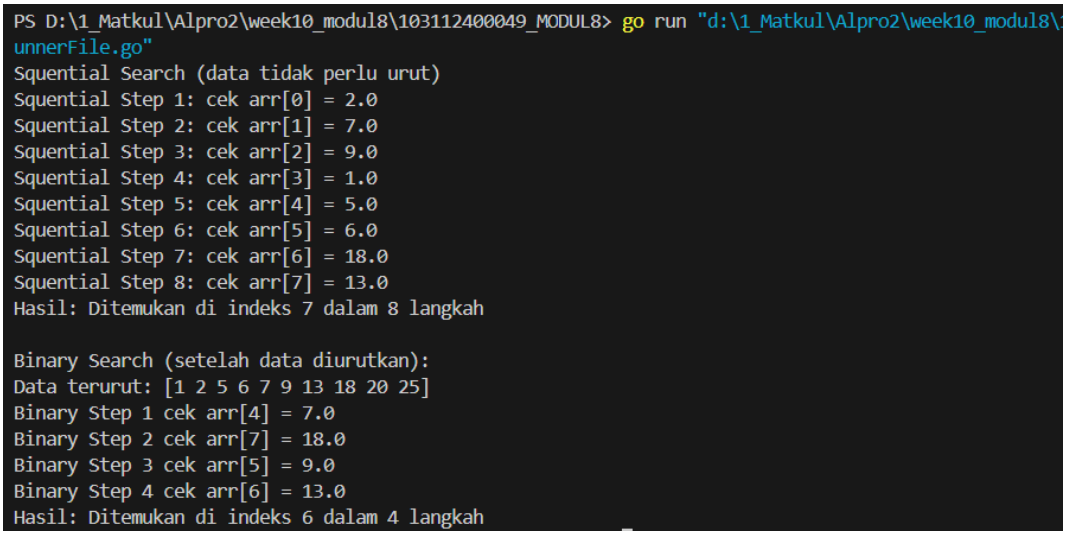
```
    target := 13.0

    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 Guided 1 ini untuk mendemonstrasikan dan membandingkan dua algoritma pencarian: pencarian sekuensial (sequential search) dan pencarian biner (binary search). Pertama, program ini mengimplementasikan fungsi sequentialSearch yang mencari sebuah elemen target dalam sebuah slice float64 dengan memeriksa setiap elemen secara berurutan, sambil mencatat jumlah iterasi. Selanjutnya, program mengimplementasikan fungsi binarySearch yang bekerja pada slice yang sudah terurut, membagi interval pencarian menjadi dua pada setiap langkahnya untuk menemukan elemen target, juga dengan pencatatan iterasi. Fungsi main kemudian menginisialisasi sebuah slice data dan nilai target, menjalankan sequentialSearch pada data awal, lalu mengurutkan data tersebut menggunakan sort.Float64s, dan terakhir menjalankan binarySearch pada data yang sudah terurut, menampilkan langkah-langkah pencarian serta hasil (indeks elemen jika ditemukan dan jumlah iterasi) untuk kedua metode tersebut.

Source code Guided 2 :

```
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)
```

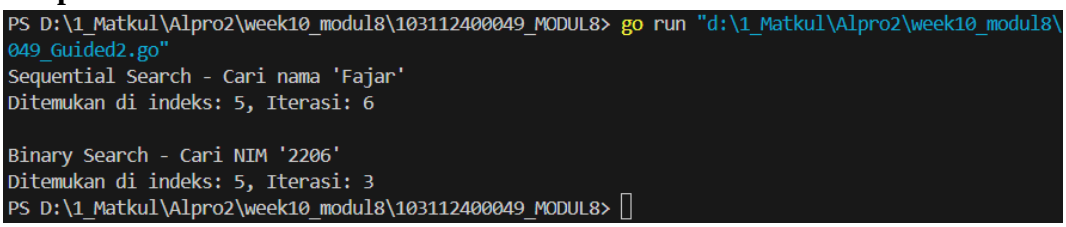
```
    }

    // 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 :



Deskripsi :

Program Guided 2 ini dirancang untuk mengelola dan mencari data mahasiswa, yang masing-masing direpresentasikan oleh sebuah struct berisi nama, NIM, kelas, jurusan, dan IPK. Program mengimplementasikan dua metode pencarian: SeqSearch_3 yang melakukan pencarian sekuensial berdasarkan nama mahasiswa pada array data mahasiswa, dan BinarySearch_3 yang melakukan pencarian biner berdasarkan NIM mahasiswa, dengan syarat data sudah terurut berdasarkan NIM. Fungsi main menginisialisasi sekumpulan data mahasiswa, kemudian mendemonstrasikan penggunaan kedua fungsi pencarian tersebut—pencarian sekuensial untuk mencari nama tertentu, diikuti dengan pengurutan data berdasarkan NIM menggunakan sort.Slice, dan kemudian pencarian biner untuk mencari NIM tertentu, sambil menampilkan indeks data yang ditemukan beserta jumlah iterasi yang diperlukan untuk setiap pencarian.

III. UNGUIDED

Source code Unguided 1 :

```
package main

import "fmt"

func main() {
    var nomorPilihan int
    suaraMasuk := 0
    suaraSah := 0
    hitungSuaraCalon := make([]int, 21)
    for {
        _, err := fmt.Scan(&nomorPilihan)
        if err != nil || nomorPilihan == 0 {
            break
        } //103112400049 Hisyam Nurdiatmoko
        suaraMasuk++
        if nomorPilihan >= 1 && nomorPilihan <= 20 {
            suaraSah++
            hitungSuaraCalon[nomorPilihan]++
        }
    }
    fmt.Printf("Suara masuk: %d\n", suaraMasuk)
    fmt.Printf("Suara sah: %d\n", suaraSah)
    for i := 1; i <= 20; i++ {
        if hitungSuaraCalon[i] > 0 {
            fmt.Printf("%d: %d\n", i, hitungSuaraCalon[i])
        }
    }
}
```

Output:

```
PS D:\1_Matkul\Alpro2\week10_modul8\103112400049_MODUL8> go run "d:\1_Matkul\Alpro2\week10_modul8\1
eRunnerFile.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:\1_Matkul\Alpro2\week10_modul8\103112400049_MODUL8>
```

Deskripsi:

Program Go Unguided 1 ini berfungsi untuk sistem penghitungan suara sederhana untuk pemilihan dengan maksimal 20 calon. Program secara terus-menerus menerima input berupa nomor pilihan calon dari pengguna hingga pengguna memasukkan angka 0 atau terjadi kesalahan input. Setiap input yang valid (antara 1 hingga 20) akan dihitung sebagai suara sah dan ditambahkan ke total suara calon yang bersangkutan, sementara semua input (termasuk yang tidak sah sebelum dihentikan oleh angka 0) dihitung sebagai suara masuk. Setelah proses input selesai, program akan menampilkan jumlah total suara masuk, jumlah total suara sah, dan rincian perolehan suara untuk setiap calon yang mendapatkan minimal satu suara.

Source code Unguided 2 :

```
package main

import "fmt"

func main() {
   var nomorPilihan int
   suaraMasuk := 0
   suaraSah := 0
   hitungSuaraCalon := make([]int, 21)

   for {
      _, err := fmt.Scan(&nomorPilihan)
      if err != nil || nomorPilihan == 0 {
         break
      }
      suaraMasuk++
      if nomorPilihan >= 1 && nomorPilihan <= 20 {
         suaraSah++
         hitungSuaraCalon[nomorPilihan]++
      }
   }
   fmt.Printf("Suara masuk: %d\n", suaraMasuk)
   fmt.Printf("Suara sah: %d\n", suaraSah)

   ketuaID := 0
   ketuaSuara := -1
   wakilID := 0
   wakilSuara := -1
   //103112400049 Hisyam Nurdiatmoko
   for id := 1; id <= 20; id++ {
      suaraCalonIni := hitungSuaraCalon[id]
      if suaraCalonIni == 0 {
         continue
      }
      if suaraCalonIni > ketuaSuara {
         wakilID = ketuaID
         wakilSuara = ketuaSuara

         ketuaID = id
         ketuaSuara = suaraCalonIni
      } else if suaraCalonIni == ketuaSuara {
         if id < ketuaID {
            wakilID = ketuaID
            wakilSuara = ketuaSuara
```

```
            ketuaID = id
        } else {
            if suaraCalonIni > wakilSuara {
                wakilID = id
                wakilSuara = suaraCalonIni
            } else if suaraCalonIni == wakilSuara {
                if id < wakilID {
                    wakilID = id
                }
            }
        }
    } else {
        if suaraCalonIni > wakilSuara {
            wakilID = id
            wakilSuara = suaraCalonIni
        } else if suaraCalonIni == wakilSuara {
            if id < wakilID {
                wakilID = id
            }
        }
    }
  }
  fmt.Printf("Ketua RT: %d\n", ketuaID)
  fmt.Printf("Wakil ketua: %d\n", wakilID)
}
```

Output:

```
PS D:\1_Matkul\Alpro2\week10_modul8\103112400049_MODUL8> go run "d:\1_Matkul\Alpro2\week10_modul8\
eRunnerFile.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:\1_Matkul\Alpro2\week10_modul8\103112400049_MODUL8>
```

Deskripsi:

Program Go Unguided 2 ini adalah sistem penghitungan suara yang tidak hanya mengakumulasi suara untuk maksimal 20 calon tetapi juga menentukan ketua dan wakil ketua RT berdasarkan hasil perolehan suara tersebut. Awalnya, program menerima input nomor pilihan calon secara berulang hingga angka 0 dimasukkan, mencatat total suara masuk dan suara sah, serta jumlah suara untuk masing-masing calon. Setelah semua suara terkumpul dan statistik awal ditampilkan, program kemudian melakukan iterasi melalui semua calon untuk mengidentifikasi dua calon dengan suara terbanyak; calon dengan suara tertinggi menjadi ketua,

dan yang tertinggi kedua menjadi wakil, dengan aturan tambahan bahwa jika terjadi perolehan suara yang sama, calon dengan nomor urut (ID) yang lebih kecil akan diutamakan untuk posisi tersebut.

Source code Unguided 3 :

```
package main

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 {
    kiri := 0
    kanan := n - 1
    indeksDitemukan := -1

    for kiri <= kanan {
        tengah := kiri + (kanan-kiri)/2
        if data[tengah] == k {
            indeksDitemukan = tengah
            break
        } else if data[tengah] < k {
            kiri = tengah + 1
        } else {
            kanan = tengah - 1
        }
    } //103112400049 Hisyam Nurdiatmoko
    return indeksDitemukan
}

func main() {
    var n, k int
    fmt.Scan(&n, &k)
    isiArray(n)
    hasilPosisi := posisi(n, k)
    if hasilPosisi != -1 {
        fmt.Println(hasilPosisi)
    } else {
        fmt.Println("TIDAK ADA")
    }
}
```

Output :

```
PS D:\1_Matkul\Alpro2\week10_modul8\103112400049_MODUL8> go run "d:\1_Matkul\Alpro2\week10_modul8\1
00049_Unguided3.go"
12 534
1 3 8 16 32 123 323 323 534 543 823 999
8
PS D:\1_Matkul\Alpro2\week10_modul8\103112400049_MODUL8> go run "d:\1_Matkul\Alpro2\week10_modul8\1
00049_Unguided3.go"
12 535
1 3 8 16 32 123 323 323 534 543 823 999
TIDAK ADA
PS D:\1_Matkul\Alpro2\week10_modul8\103112400049_MODUL8>
```

Deskripsi :

Program Go Unguided 3 ini digunakan untuk mencari posisi sebuah angka tertentu (k) dalam sebuah array integer (data) yang berukuran hingga NMAX (1.000.000 elemen) menggunakan algoritma pencarian biner. Fungsi main terlebih dahulu meminta pengguna memasukkan jumlah elemen (n) yang akan diisi ke dalam array dan angka (k) yang ingin dicari. Kemudian, fungsi isiArray dipanggil untuk mengisi n elemen array data dengan input dari pengguna. Setelah itu, fungsi posisi melakukan pencarian biner pada n elemen pertama array tersebut untuk menemukan k; jika k ditemukan, fungsi mengembalikan indeksnya, dan jika tidak, mengembalikan -1. Akhirnya, program mencetak indeks yang ditemukan atau pesan "TIDAK ADA" jika elemen tidak terdapat dalam array, dengan asumsi penting bahwa data yang dimasukkan oleh pengguna ke dalam array sudah dalam keadaan terurut agar pencarian biner memberikan hasil yang benar.

IV. KESIMPULAN

Praktikum Modul 8 mengenai "Pencarian Nilai Acak pada Himpunan Data" ini telah berhasil mendemonstrasikan implementasi dan perbandingan dua teknik pencarian utama, yaitu *Sequential Search* dan *Binary Search*, serta aplikasinya pada berbagai tipe data termasuk *struct* dalam bahasa pemrograman Go. Melalui latihan terbimbing (*Guided*) dan mandiri (*Unguided*), praktikan dapat memahami perbedaan fundamental antara kedua algoritma tersebut, di mana *Sequential Search* melakukan pencarian secara linear elemen per elemen tanpa memerlukan data terurut, sementara *Binary Search* bekerja secara efisien dengan membagi interval pencarian pada data yang sudah terurut. Implementasi pada kasus nyata seperti pencarian data mahasiswa berdasarkan nama (menggunakan *Sequential Search*) dan NIM (menggunakan *Binary Search* setelah pengurutan), serta simulasi penghitungan suara hingga penentuan ketua dan wakil RT, menunjukkan penerapan praktis dari konsep-konsep yang dipelajari. Selain itu, praktikum ini juga menekankan pentingnya prasyarat data terurut untuk efektivitas *Binary Search* dan bagaimana menangani pencarian pada tipe data kompleks seperti *struct* dengan memilih field yang sesuai sebagai kunci pencarian. Secara keseluruhan, praktikum ini memberikan pemahaman yang komprehensif mengenai algoritma pencarian dan implementasinya dalam menyelesaikan masalah-masalah komputasional.

V. REFERENSI

MODUL 8 Algoritma Pemrograman 2