LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL 8

PENCARIAN NILAI ACAK PADA HIMPUNAN DATA

Oleh:

RYAN AKEYLA NOVIANTO WIDODO

103112400081

12 IF 01

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

I. DASAR TEORI

Dasar teori pencarian nilai acak pada himpunan data dalam konteks coding Go bergantung pada bagaimana himpunan data tersebut direpresentasikan dan bagaimana "acak" didefinisikan. Ada beberapa pendekatan, dan pilihan terbaik bergantung pada kebutuhan spesifik Anda.

1. Representasi Himpunan Data:

- Array/Slice: Struktur data paling sederhana di Go. Pencarian nilai acak di sini mudah jika Anda mendefinisikan "acak" sebagai pemilihan indeks secara acak.
- Map: Cocok jika Anda perlu mengakses elemen berdasarkan kunci. Pencarian nilai acak di sini memerlukan iterasi melalui keys dan pemilihan acak salah satunya.
- Struktur Data Khusus (misalnya, tree, graph): Jika himpunan data besar dan terstruktur, struktur data khusus bisa lebih efisien, tetapi pencarian acak akan memerlukan algoritma yang lebih kompleks.

2. Definisi "Acak":

- Acak Benar (True Randomness): Membutuhkan generator angka acak yang benar-benar acak (misalnya, menggunakan sumber entropi dari sistem operasi). Go menyediakan paket crypto/rand untuk ini.
- Acak Semu (Pseudorandomness): Lebih umum dan efisien. Menggunakan algoritma deterministik untuk menghasilkan urutan angka yang tampak acak. Go menyediakan paket math/rand .

3. Algoritma Pencarian Nilai Acak

II. GUIDED

Source Code + Screenshot hasil program beserta penjelasan

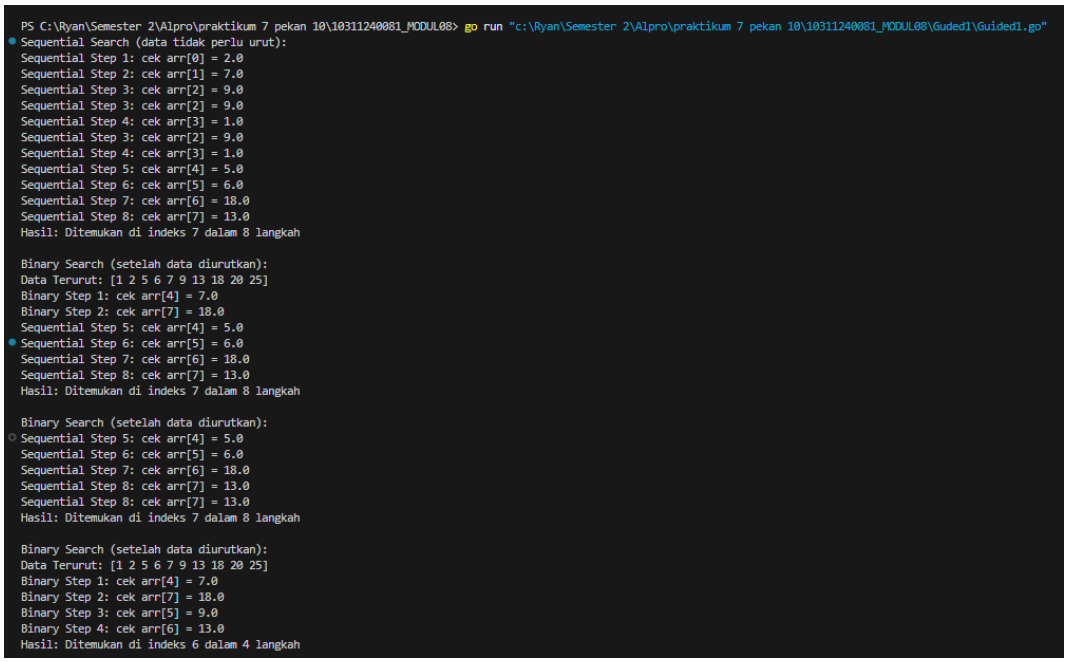
Guided 1

Coding:

```
package main

import (
   "fmt"
   "sort"
)

func sequentialSeacrh(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 perlu urut):")
    idxSeq, iterSeq := sequentialSeacrh(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 dalam %d langkah\n\n", iterSeq)
    }

    // Binary search perlu array diurutkan
    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\n", idxBin, iterBin)
    } else {
        fmt.Printf("Hasil: Tidak Ditemukan setelah indeks %d langkah\n", iterBin)
    }
}
```

Hasil Coding:



Tujuan program ini adalah membuat program yang mana kita membandingkan performa dua algoritma pencarian: Sequential Search dan Binary Search.
Program ini mendemonstrasikan bagaimana kedua algoritma bekerja, langkah demi langkah, dengan menampilkan indeks yang diperiksa pada setiap iterasi. Hal ini memungkinkan kita untuk melihat perbedaan efisiensi antara kedua algoritma.

Guided 2

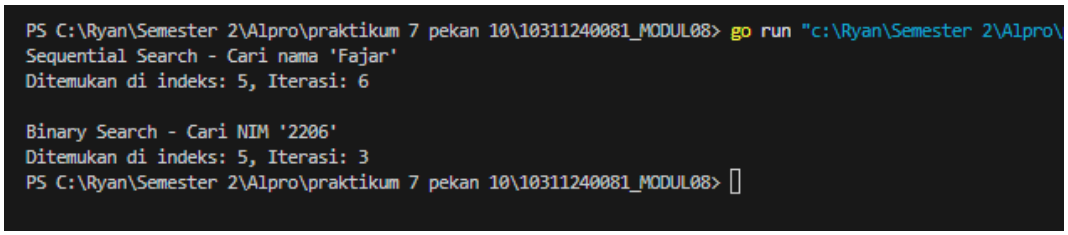
Coding:

```
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
    temp := [10]mahasiswa{
        {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},
    }

    for i := 0; i < n; i++ {
        data[i] = temp[i]
    }

    // 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)
    }
}
```

Hasil Coding:



Tujuan program ini adalah membuat program yang mana kita membandingkan performa algoritma pencarian Sequential Search dan Binary Search pada struktur data berupa array dari tipe data mahasiswa . Program ini menunjukkan perbedaan implementasi dan efisiensi kedua algoritma tersebut dalam konteks pencarian data berdasarkan atribut tertentu dari struktur mahasiswa .

III. UNGUIDED

Source Code + Screenshot hasil program beserta penjelasan

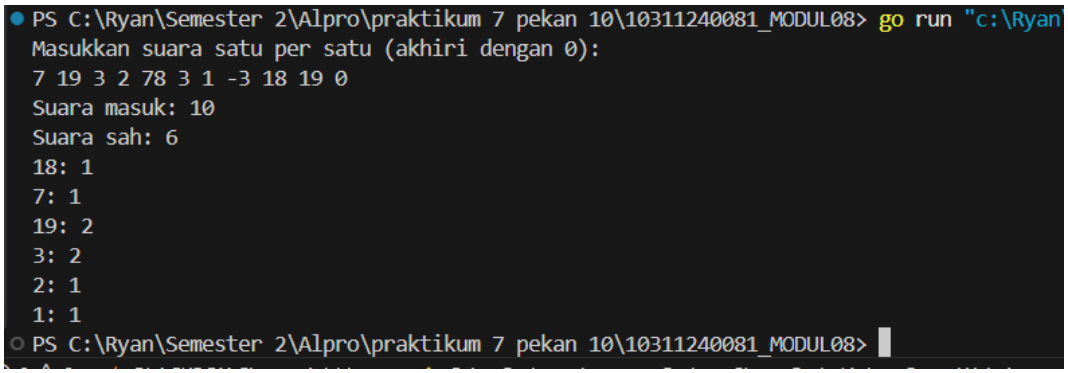
Guided 1

Coding :

```
package main

import (
    "fmt"
    "sort"
)

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

func main() {
    validCandidates := make([]int, 20)
    for i := range validCandidates {
        validCandidates[i] = i + 1
    }
    sort.Ints(validCandidates)

    voteCounts := make(map[int]int)
    totalVotes := 0
    var input int

    fmt.Println("Masukkan suara satu per satu (akhiri dengan 0):")
    for {
        _, err := fmt.Scan(&input)
        if err != nil || input == 0 {
            break
```

```
        }
        totalVotes++
        index := binarySearch(validCandidates, input)
        if index != -1 {
            voteCounts[input]++
        }
    }

    fmt.Printf("Suara masuk: %d\n", totalVotes)
    fmt.Printf("Suara sah: %d\n", len(voteCounts))

    for candidate, count := range voteCounts {
        fmt.Printf("%d: %d\n", candidate, count)
    }
}
```

Hasil Coding :



Tujuan program ini adalah membuat program yang mana kita menghitung suara dalam sebuah pemilihan.

Guided 2

Coding :

```
package main

import (
    "fmt"
    "sort"
)

func binarySearch(arr []int, target int) bool {
```

```
    low := 0
    high := 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
}

type Calon struct {
    Nomor int
    Suara int
}

func main() {
    var input int
    validCandidates := make([]int, 20)
    for i := 0; i < 20; i++ {
        validCandidates[i] = i + 1
    }
    sort.Ints(validCandidates)

    voteCounts := make([]int, 21)
    totalVotes := 0
    validVotes := 0

    fmt.Println("Masukkan suara satu per satu (akhiri dengan 0):")
    for {
        _, err := fmt.Scan(&input)
        if err != nil {
            break
        }
        if input == 0 {
            break
        }
        totalVotes++
        if binarySearch(validCandidates, input) {
            voteCounts[input]++
            validVotes++
        }
```

```
    }
    var hasil []Calon
    for i := 1; i <= 20; i++ {
        if voteCounts[i] > 0 {
            hasil = append(hasil, Calon{Nomor: i, Suara: voteCounts[i]})
        }
    }
    sort.Slice(hasil, func(i, j int) bool {
        if hasil[i].Suara == hasil[j].Suara {
            return hasil[i].Nomor < hasil[j].Nomor
        }
        return hasil[i].Suara > hasil[j].Suara
    })

    fmt.Printf("Suara masuk: %d\n", totalVotes)
    fmt.Printf("Suara sah: %d\n", validVotes)

    if len(hasil) == 0 {
        fmt.Println("Tidak ada suara sah, tidak ada ketua dan wakil.")
    } else if len(hasil) == 1 {
        fmt.Printf("Ketua RT: Calon nomor %d\n", hasil[0].Nomor)
        fmt.Println("Wakil RT: Tidak tersedia (hanya satu calon sah).")
    } else {
        fmt.Printf("Ketua RT: Calon nomor %d\n", hasil[0].Nomor)
        fmt.Printf("Wakil RT: Calon nomor %d\n", hasil[1].Nomor)
    }
}
```

Hasil Coding :

```
PS C:\Ryan\Semester 2\Alpro\praktikum 7 pekan 10\10311240081_MODUL08> go run "c:\Ryan\Semester 2\Alpr
Masukkan suara satu per satu (akhiri dengan 0):
7 19 3 2 78 3 1 -3 18 19 0
Suara masuk: 10
Suara sah: 8
Ketua RT: Calon nomor 3
Wakil RT: Calon nomor 19
PS C:\Ryan\Semester 2\Alpro\praktikum 7 pekan 10\10311240081_MODUL08>
```

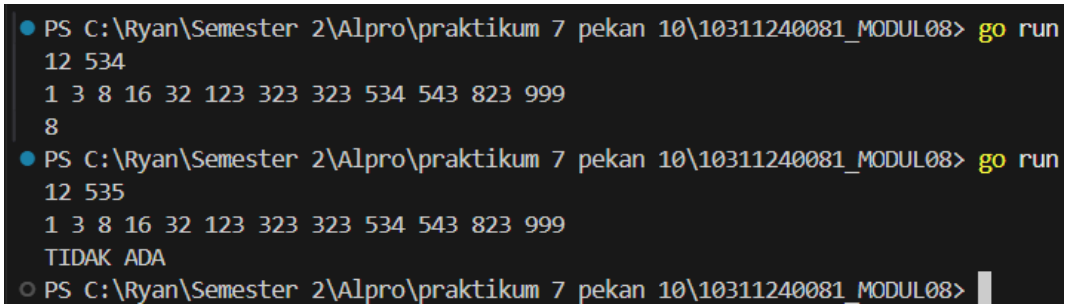
Tujuan program ini adalah membuat program yang mana kita memprogram penghitung suara pemilihan Ketua dan Wakil Ketua RT. Program ini memberikan implementasi yang relatif sederhana dan mudah dipahami untuk penghitungan suara pemilihan Ketua dan Wakil Ketua RT.

Guided 3

Coding :

```
package main

import "fmt"

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

    data := make([]int, n)
    for i := range data {
        fmt.Scan(&data[i])
    }

    pos := findPosition(data, k)

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

func findPosition(arr []int, target int) int {
    for i, val := range arr {
        if val == target {
            return i
        }
    }
    return -1
}
```

Hasil Coding :



Tujuan program ini adalah membuat program yang mana kita mencari posisi (index) dari suatu angka tertentu dalam sebuah array. Program ini bisa ditingkatkan dengan menggunakan algoritma pencarian yang lebih efisien, seperti binary search, jika array telah terurut. Binary search memiliki kompleksitas waktu O(log n) dibandingkan dengan linear search yang memiliki kompleksitas O(n). Namun, perlu diingat bahwa binary search hanya dapat digunakan pada array yang telah terurut.

IV. KESIMPULAN

Kesimpulannya, pencarian nilai acak dalam himpunan data di Go bergantung pada representasi data (array, map, dll.) dan tingkat keacakan yang dibutuhkan (acak semu atau acak benar). Kode yang efisien dan tepat memerlukan pemilihan fungsi dan algoritma yang sesuai dengan kebutuhan tersebut, dengan memperhatikan inisialisasi seed untuk generator angka acak dan potensi penggunaan teknik seperti reservoir sampling untuk himpunan data yang sangat besar.
Paket math/rand umumnya cukup untuk sebagian besar kasus, sementara crypto/rand dibutuhkan untuk aplikasi yang memerlukan keacakan yang lebih tinggi.

V. REFERENSI

Modul 8 Pencarian Nilai Acak Pada Himpunan Data, Algoritma Pemrograman 2

https://dasarpemrogramangolang.novalagung.com/