LAPORAN PRAKTIKUM
ALGORITMA PEMOGRAMAN II

MODUL XI

PENCARIAN NILAI ACAK PADA HIMPUNAN DATA

Oleh:

NAMA: SHEILA DWI YULIANA

NIM: 2311102292

KELAS: IF-11-07

S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

UNIVERSITY TELKOM PURWOKERTO

2024

A. DASAR TEORI

1. Definisi Himpunan Data

   Himpunan data adalah sekumpulan elemen atau entitas yang merepresentasikan informasi. Data dapat berbentuk angka, teks, objek, atau kombinasi dari berbagai jenis. Dalam ilmu komputer, himpunan data sering dikelola menggunakan struktur data seperti array, daftar, atau tabel.

2. Pengacakan (Randomization)

   Pengacakan adalah proses memilih elemen secara acak dari suatu himpunan. Elemen yang dipilih memiliki peluang yang sama (uniform probability) atau sesuai dengan distribusi tertentu. Pengacakan sering digunakan untuk menghindari bias atau untuk membuat simulasi, eksperimen, atau pengambilan sampel lebih representatif.

3. Konsep Nilai Acak

   Nilai acak adalah elemen yang dipilih tanpa pola tertentu dari suatu himpunan data. Dalam matematika, ini dilakukan dengan memanfaatkan fungsi distribusi probabilitas. Dalam pemrograman, nilai acak biasanya dihasilkan dengan fungsi random yang tersedia di pustaka bahasa pemrograman.

4. Teknik Pencarian Nilai Acak pada Himpunan Data

   Untuk mencari nilai acak dalam himpunan data, langkah-langkah yang biasa dilakukan meliputi:

   1. Identifikasi Ruang Sampel

      Ruang sampel didefinisikan sebagai semua elemen dalam himpunan. Misalnya, jika himpunan adalah {1, 2, 3, 4, 5}, maka ruang sampelnya adalah 5 elemen.

   2. Penggunaan Fungsi Acak

      Fungsi acak digunakan untuk menghasilkan indeks atau elemen acak. Contoh: Dalam Python, random.choice(himpunan) memilih elemen acak dari suatu himpunan.

   3. Pemilihan Berdasarkan Distribusi

      Elemen dapat dipilih dengan distribusi seragam (uniform) atau distribusi lainnya, seperti normal atau eksponensial.

   4. Iterasi dan Validasi

Jika pemilihan bergantung pada kondisi tertentu, nilai acak dipilih berulang kali hingga kondisi terpenuhi.

5. Aplikasi dalam Dunia Nyata
   Sampling Statistik: Mengambil sampel acak dari populasi untuk analisis.
   Simulasi Monte Carlo: Penggunaan pengacakan untuk mensimulasikan sistem yang kompleks.
   Pemilihan Konten: Algoritma acak digunakan untuk merekomendasikan konten kepada pengguna, seperti di YouTube atau Spotify.
   Keamanan Kriptografi: Pengacakan elemen dalam algoritma enkripsi.

6. Faktor-Faktor yang Perlu Dipertimbangkan
   Ukuran Himpunan Data
   Semakin besar himpunan, semakin kompleks pemrosesan dan waktu pencarian.

   Kualitas Generator Angka Acak (Random Number Generator)
   Kualitas angka acak sangat penting, terutama dalam aplikasi keamanan.

   Kompleksitas Algoritma
   Algoritma yang digunakan harus efisien untuk menangani himpunan besar.

7. Algoritma Pengacakan Sederhana
   Misalkan ada himpunan data S = {a1, a2, ..., an}.
   1. Tentukan ukuran himpunan n.
   2. Gunakan fungsi acak untuk menghasilkan indeks i di antara 1 hingga n.
   3. Pilih elemen S[i] sebagai elemen acak.

B. GUIDED

1. Guided 1

```go
package main
import (
    "fmt"
)
// Definisi struct mahasiswa
type mahasiswa struct {
    nama    string
    nim     string
    kelas   string
    jurusan string
    ipk     float64
}

// Definisi array bertipe struct mahasiswa
type arrMhs [2023]mahasiswa

// Sequential Search berdasarkan nama
func SeqSearch_3(T arrMhs, n int, X string) int {
    /* Mengembalikan indeks mahasiswa dengan nama X,
       atau -1 apabila tidak ditemukan pada array T
       yang berisi n data mahasiswa */
    var found int = -1
    var j int = 0
    for j < n && found == -1 {
        if T[j].nama == X {
            found = j
        }
        j = j + 1
    }
    return found
}
```

```go
// Binary Search berdasarkan nim
func BinarySearch_3(T arrMhs, n int, X string) int {
    /* Mengembalikan indeks mahasiswa dengan nim X,
       atau -1 apabila tidak ditemukan pada array T
       yang berisi n data mahasiswa dan terurut membesar berdasarkan nim */
    var found int = -1
    var med int
    var kr int = 0
    var kn int = n - 1

    for kr <= kn && found == -1 {
        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
}
```

```go
// Fungsi utama untuk demonstrasi
func main() {
    var data arrMhs
    var n int

    // Input jumlah mahasiswa
    fmt.Print("Masukkan jumlah mahasiswa: ")
    fmt.Scanln(&n)

    // Input data mahasiswa
    fmt.Println("Masukkan data mahasiswa:")
    for i := 0; i < n; i++ {
        fmt.Printf("Mahasiswa %d\n", i+1)
        fmt.Print("Nama: ")
        fmt.Scanln(&data[i].nama)
        fmt.Print("NIM: ")
        fmt.Scanln(&data[i].nim)
        fmt.Print("Kelas: ")
        fmt.Scanln(&data[i].kelas)
        fmt.Print("Jurusan: ")
        fmt.Scanln(&data[i].jurusan)
        fmt.Print("IPK: ")
        fmt.Scanln(&data[i].ipk)
    }
```

```go
    // Pencarian berdasarkan nama
    var cariNama string
    fmt.Print("Masukkan nama mahasiswa yang ingin dicari: ")
    fmt.Scanln(&cariNama)
    posNama := SeqSearch_3(data, n, cariNama)
    if posNama == -1 {
        fmt.Println("Mahasiswa dengan nama tersebut tidak ditemukan.")
    } else {
        fmt.Printf("Mahasiswa ditemukan pada indeks: %d\n", posNama)
    }

    // Pencarian berdasarkan nim
    var cariNIM string
    fmt.Print("Masukkan NIM mahasiswa yang ingin dicari: ")
    fmt.Scanln(&cariNIM)
    posNIM := BinarySearch_3(data, n, cariNIM)
    if posNIM == -1 {
        fmt.Println("Mahasiswa dengan NIM tersebut tidak ditemukan.")
    } else {
        fmt.Printf("Mahasiswa ditemukan pada indeks: %d\n", posNIM)
    }
}
```

Output:

```
Masukkan jumlah mahasiswa: 3
Masukkan data mahasiswa:
Mahasiswa 1
Nama: Ningning
NIM: 2371537252
Kelas: 02
Jurusan: DKV
IPK: 3.8
Mahasiswa 2
Nama: Esther
NIM: 2381646373
Kelas: 02
Jurusan: DKV
IPK: 3.9
Mahasiswa 3
Nama: Hanni
NIM: 2334618362
Kelas: 02
Jurusan: DKV
IPK: 3.8
Masukkan nama mahasiswa yang ingin dicari: Hanni
Mahasiswa ditemukan pada indeks: 2
Masukkan NIM mahasiswa yang ingin dicari: 2381646373
Mahasiswa ditemukan pada indeks: 1
```

Penjelasan code:

Kode pada atas merupakan acara pada bahasa Go buat mengelola data mahasiswa memakai struktur data array & melakukan pencarian data memakai 2 metode, yaitu Sequential Search dari nama & Binary Search dari NIM.

Program dimulai menggunakan mendefinisikan struct 'mahasiswa' buat menyimpan atribut misalnya nama, NIM, kelas, jurusan, & IPK, dan array 'arrMhs' buat menyimpan sampai 2023 data mahasiswa.

Fungsi 'SeqSearch_3' mencari mahasiswa dari nama menggunakan pencarian linear sampai data ditemukan atau semua elemen diperiksa, sementara 'BinarySearch_3' mencari dari NIM dalam array yg telah terurut memakai metode pembagian interval.

Dalam fungsi utama ('main'), acara meminta pengguna buat memasukkan jumlah mahasiswa, data setiap mahasiswa, kemudian menaruh opsi buat mencari mahasiswa dari nama atau NIM, menggunakan output berupa indeks data yg ditemukan atau pesan bahwa data nir ditemukan.

2. Guided 2

```go
package main

import "fmt"
// Define arrInt as an array of integers with a fixed size
type arrInt [4321]int
// Binary search function for descending sorted array
func BinarySearch_2(T arrInt, n int, X int) int {
    /* Mengembalikan indeks dari X apabila X ditemukan di dalam array T
       yang berisi n buah bilangan bulat terurut secara descending/menciut,
       atau -1 apabila X tidak ditemukan */
    var found int = -1
    var med int
    var kr int = 0
    var kn int = n - 1

    // Iterasi mencari elemen dengan binary search
    for kr <= kn && found == -1 {
        med = (kr + kn) / 2
        if X > T[med] { // Karena descending, jika X lebih besar, bergerak ke kiri
            kn = med - 1
        } else if X < T[med] { // Jika X lebih kecil, bergerak ke kanan
            kr = med + 1
        } else { // Jika ditemukan
            found = med
        }
    }
    return found
}
```

```go
func main() {
    // Contoh penggunaan
    var data arrInt
    var n int
    // Input jumlah elemen
    fmt.Print("Masukkan jumlah elemen dalam array: ")
    fmt.Scanln(&n)
    // Input elemen array (harus terurut secara descending)
    fmt.Println("Masukkan elemen array (harus terurut menurun):")
    for i := 0; i < n; i++ {
        fmt.Printf("Elemen %d: ", i+1)
        fmt.Scanln(&data[i])
    }

    // Input elemen yang dicari
    var search int
    fmt.Print("Masukkan elemen yang ingin dicari: ")
    fmt.Scanln(&search)

    // Panggil fungsi binary search
    result := BinarySearch_2(data, n, search)

    // Cetak hasil
    if result == -1 {
        fmt.Println("Elemen tidak ditemukan dalam array.")
    } else {
        fmt.Printf("Elemen ditemukan pada indeks: %d\n", result)
    }
}
```

Output:

```
Masukkan jumlah elemen dalam array: 5
Masukkan elemen array (harus terurut menurun):
Elemen 1: 22
Elemen 2: 15
Elemen 3: 9
Elemen 4: 5
Elemen 5: 2
Masukkan elemen yang ingin dicari: 9
Elemen ditemukan pada indeks: 2
```

Penjelasan kode:

Kode pada atas merupakan implementasi acara pada bahasa Go buat mencari sebuah elemen pada array sapta bundar yg telah diurutkan secara menurun (descending) memakai metode Binary Search.

Fungsi 'BinarySearch_2' mendapat array 'T', jumlah elemen 'n', & elemen yg dicari 'X', kemudian mengembalikan indeks elemen tadi bila ditemukan atau '-1' bila nir ditemukan.
Algoritma memanfaatkan pembagian interval menggunakan membandingkan elemen tengah 'med' terhadap 'X': bila 'X' lebih besar, pencarian berkecimpung ke kiri, & bila lebih kecil, pencarian berkecimpung ke kanan.
Program meminta pengguna memasukkan jumlah elemen, nilai elemen yg telah diurutkan menurun, & elemen yg akan dicari.
Hasil pencarian ditampilkan menjadi indeks bila elemen ditemukan atau pesan bahwa elemen nir terdapat pada array.

3. Guided 3

```go
package main
import "fmt"
// Define arrStr as an array of strings with a fixed size
type arrStr [1234]string
// Sequential search function
func SeqSearch_1(T arrStr, n int, X string) int {
    /* Mengembalikan indeks dari X apabila X ditemukan di dalam array T
       yang berisi n buah teks, atau -1 apabila X tidak ditemukan */
    var found int = -1
    var j int = 0

    // Iterasi mencari elemen yang cocok
    for j < n && found == -1 {
        if T[j] == X {
            found = j
        }
        j = j + 1
    }
    return found
}
```

```go
func main() {
    // Contoh penggunaan
    var data arrStr
    var n int

    // Input jumlah elemen
    fmt.Print("Masukkan jumlah elemen dalam array: ")
    fmt.Scanln(&n)

    // Input elemen array
    fmt.Println("Masukkan elemen array:")
    for i := 0; i < n; i++ {
        fmt.Printf("Elemen %d: ", i+1)
        fmt.Scanln(&data[i])
    }
    // Input elemen yang dicari
    var search string
    fmt.Print("Masukkan elemen yang ingin dicari: ")
    fmt.Scanln(&search)
    // Panggil fungsi sequential search
    result := SeqSearch_1(data, n, search)
    // Cetak hasil
    if result == -1 {
        fmt.Println("Elemen tidak ditemukan dalam array.")
    } else {
        fmt.Printf("Elemen ditemukan pada indeks: %d\n", result)
    }
}
```

Output:

```
Masukkan jumlah elemen dalam array: 5
Masukkan elemen array:
Elemen 1: 7
Elemen 2: 18
Elemen 3: 25
Elemen 4: 28
Elemen 5: 30
Masukkan elemen yang ingin dicari: 7
Elemen ditemukan pada indeks: 0
```

Penjelasan kode:

Kode pada atas merupakan implementasi acara pada bahasa Go buat mencari elemen teks eksklusif pada array memakai metode Sequential Search.
Array 'arrStr' didefinisikan buat menyimpan sampai 1234 elemen bertipe string.

Fungsi 'SeqSearch_1' melakukan pencarian linear menggunakan mengusut setiap elemen pada array sampai elemen yg dicari ('X') ditemukan, kemudian mengembalikan indeksnya, atau '-1' bila nir ditemukan.
Dalam fungsi utama ('main'), acara meminta pengguna buat memasukkan jumlah elemen, isi array berupa string, & elemen yg ingin dicari.
Hasil pencarian ditampilkan menggunakan menuliskan indeks elemen bila ditemukan atau pesan bahwa elemen nir terdapat pada array.

C. UNGUIDED

1. Unguided 1

```go
package main

import "fmt"

func main() {
    var suaraTotal, suaraValid int
    var frekuensi [21]int // Array untuk menghitung frekuensi setiap calon

    fmt.Println("Masukkan pilihan suara (0 untuk berhenti):")

    // Input data
    for {
        var pilihan int
        fmt.Scan(&pilihan)
        if pilihan == 0 {
            break
        }
        if pilihan >= 1 && pilihan <= 20 {
            suaraTotal++
            suaraValid++
            frekuensi[pilihan]++
        }
    }

    // Output hasil
    fmt.Println("Jumlah suara total:", suaraTotal)
    fmt.Println("Jumlah suara valid:", suaraValid)

    fmt.Println("Hasil perhitungan suara:")
    for calon, jumlahSuara := range frekuensi[1:] {
        if jumlahSuara > 0 {
            fmt.Printf("Calon %d: %d suara\n", calon+1, jumlahSuara)
        }
    }
}
```

Output:

```
Masukkan pilihan suara (0 untuk berhenti):
7 19 3 2 78 3 1 -3 18 19 0
Jumlah suara total: 8
Jumlah suara valid: 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
```

Penjelasan code:

Program ini merupakan sebuah pelaksanaan penghitungan bunyi buat pemilihan calon menggunakan memakai input menurut pengguna.

Pertama, acara meminta pengguna buat memasukkan pilihan bunyi mereka, pada mana bunyi yg valid berada pada antara 1 sampai 20 (termasuk).

Setiap kali pengguna memasukkan bunyi yg valid, jumlah total bunyi & bunyi valid dihitung, dan frekuensi bunyi buat setiap calon dicatat pada array.

Program akan terus mendapat input sampai pengguna memasukkan nomor 0 buat berhenti.

Setelah itu, acara menampilkan jumlah total bunyi, jumlah bunyi valid, & output perhitungan bunyi buat setiap calon yg memperoleh bunyi, dari frekuensi yg tercatat.

2. Unguided 2

```go
package main

import (
	"fmt"
	"sort"
)

type Calon struct {
	nomor    int
	jumlahSuara int
}

func main() {
	var suaraTotal, suaraValid int
	var suaraCalon []Calon

	fmt.Println("Masukkan pilihan suara (0 untuk berhenti):")

	// Input data
	for {
		var pilihan int
		fmt.Scan(&pilihan)
		if pilihan == 0 {
			break
		}
		if pilihan >= 1 && pilihan <= 20 {
			suaraTotal++
			suaraValid++

			// Cari calon dalam slice, jika sudah ada, tambah suaranya
			found := false
			for i := range suaraCalon {
				if suaraCalon[i].nomor == pilihan {
```

```
                // Cari calon dalam slice, jika sudah ada, tambah suaranya
                found := false
                for i := range suaraCalon {
                        if suaraCalon[i].nomor == pilihan {
                                suaraCalon[i].jumlahSuara++
                                found = true
                                break
                        }
                }

                // Jika calon belum ada, tambahkan ke slice
                if !found {
                        suaraCalon = append(suaraCalon, Calon{nomor: pilihan, jumlahSuara: 1})
                }
        }
    }

    // Urutkan calon berdasarkan jumlah suara (descending) dan nomor calon (ascending)
    sort.Slice(suaraCalon, func(i, j int) bool {
        if suaraCalon[i].jumlahSuara == suaraCalon[j].jumlahSuara {
            return suaraCalon[i].nomor < suaraCalon[j].nomor
        }
        return suaraCalon[i].jumlahSuara > suaraCalon[j].jumlahSuara
    })

    // Output hasil
    fmt.Println("Jumlah suara total:", suaraTotal)
    fmt.Println("Jumlah suara valid:", suaraValid)
```

```
    // Output hasil
    fmt.Println("Jumlah suara total:", suaraTotal)
    fmt.Println("Jumlah suara valid:", suaraValid)

    fmt.Println("Hasil perhitungan suara:")
    fmt.Printf("Ketua RT: Calon %d\n", suaraCalon[0].nomor)
    fmt.Printf("Wakil Ketua RT: Calon %d\n", suaraCalon[1].nomor)
}
```

Output:

```
Masukkan pilihan suara (0 untuk berhenti):
7 19 3 2 78 3 1 -3 18 19 0
Jumlah suara total: 8
Jumlah suara valid: 8
Hasil perhitungan suara:
Ketua RT: Calon 3
Wakil Ketua RT: Calon 19
```

Penjelasan kode:

Program ini menghitung & mengurutkan output pemilihan bunyi buat calon kepala & wakil kepala RT.

Pengguna diminta buat memasukkan pilihan bunyi mereka (angka calon antara 1 sampai 20) sampai mereka memasukkan nomor 0 buat berhenti.

Setiap bunyi valid yg dimasukkan akan dihitung, & data bunyi calon akan disimpan pada slice 'suaraCalon' yg berisi struktur 'Calon' menggunakan 2 atribut: 'angka ' calon & 'jumlahSuara'.
Program mencari apakah calon telah terdapat pada slice; apabila terdapat, jumlah suaranya akan bertambah, apabila tidak, calon baru akan ditambahkan.
Setelah input selesai, slice 'suaraCalon' diurutkan menurut jumlah bunyi secara menurun & angka calon secara menaik.
Terakhir, acara menampilkan jumlah total bunyi, jumlah bunyi valid, & mencetak output perhitungan bunyi, menggunakan calon menggunakan bunyi terbanyak menjadi kepala RT & calon ke 2 terbanyak menjadi wakil kepala RT.

3. Unguided 3

```go
package main

import "fmt"

const NMAX = 1000000
var data [NMAX]int

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

	// Isi array dengan data
	isiArray(n)

	// Cari posisi k
	posisiK := posisi(n, k)

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

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

```go
func posisi(n, k int) int {
        // Pencarian biner
        left, right := 0, n-1
        for left <= right {
                mid := (left + right) / 2
                if data[mid] == k {
                        return mid
                } else if data[mid] < k {
                        left = mid + 1
                } else {
                        right = mid - 1
                }
        }
        return -1
}
```

Output:

```
12 534
1 3 8 16 32 123 323 323 534 543 823 999
8
PS C:\Users\INTEL i3> go run "C:\Users\I
12
535
1
3
8
16
32
123
323
323
534
543
823
999
TIDAK ADA
```

Penjelasan program:

Program ini mencari posisi suatu elemen pada array memakai prosedur pemecahan pencarian biner.

Pertama, acara membaca 2 input yaitu 'n' (jumlah elemen pada array) dan 'k' (elemen yg dicari).

Kemudian, array 'data' diisi dengan 'n' elemen yg dimasukkan sang pengguna melalui fungsi 'isiArray()'.

Fungsi 'posisi()' lalu melakukan pencarian biner buat menemukan posisi elemen 'k' pada array yg telah terurut.

apabila elemen ditemukan, fungsi tadi mengembalikan indeksnya, bila nir, mengembalikan -1.
Hasil pencarian lalu dicetak, dengan "TIDAK ADA" bila elemen nir ditemukan.
Program mengasumsikan bahwa array telah terurut sebelum pencarian dilakukan.