LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL XI

PENCARIAN NILAI ACAK PADA HIMPUNAN DATA

Oleh:

FEROS PEDROSA VALENTINO

103112400055

IF-12-01

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

I. DASAR TEORI

Selain pencarian nilai ekstrim, kita terkadang juga mencari suatu data yang lebih spesifik. Misalnya mencari mahasiswa dengan nama atau nim tertentu, mencari rumah dengan alamat tertentu, dan lainnya. Berbeda dengan pencarian nilai ekstrim, yang mana nilai yang dicari selalu ditemukan, maka pada kasus pencarian ini terdapat kemungkinan bahwa data yang dicari tidak ditemukan. Selain itu pada kasus pencarian ini akan diperkenal algoritma pencarian yang memanfaatkan keterurutan data.

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.

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 ***kr*** s.d. kanan ***kn***. 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.

Algoritma ini dikenal dengan nama Binary Search.

Sebagai catatan algoritma binary search untuk array terurut membesar (ascending) akan berbeda dengan terurut mengecil (descending), sehingga algoritma ini tidak akan berjalan apabila terbalik. Misalnya array terurut secara membesar, tetapi algoritma binary search yang digunakan untuk array terurut mengecil. Hal ini mengakibatkan pencarian binary search tidak akan berhasil. Proses binary search akan berakhir apabila nilai kr > kn (data tidak ditemukan) atau found sudah tidak bernilai false atau -1 (data ditemukan).

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

1. Guided1

Source code:

```
//Feros Pedrosa

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() {
    //Array awal
    data := []float64{2, 7, 9, 1, 5, 6, 18, 13, 25, 20}
    target := 13.0

    fmt.Println("sequentialSearch (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)
    }

    //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("Haasil: Ditemukan di indeks %d dalam %d
langkah", idxBin, iterBin)
    } else {
        fmt.Printf("Hasil: Tidak ditemukn setelah %d
langkah\n", iterBin)
    }
}
```

Output:

```
PS C:\ALPRO\semester2_alpro2\alpro2_sequence> go run "c:\ALPRO\semester2_alpro2\alpro2_sequence\guided1\guided1.go"
sequentialSearch (data tidak perlu urut):
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
Haasil: Ditemukan di indeks 6 dalam 4 langkah
```

Deskripsi Program:
Program diatas ditulis dalam bahasa go dan berfungsi untuk melakukan pencarian elemen dalam sebuah array menggunakan dua metode yaitu sekuensial dan biner. Program mengimpor paket yang diperlukan yaitu fmt untuk format output dan sort untuk mengurutkan array. Fungsi sequentialSearch menerima array bertipe float64 dan nilai target yang ingin dicari. Proses pencarian dilakukan dengan memeriksa setiap elemen dalam array secara berurutan. Saat setiap elemen diperiksa, program akan mencetak langkah yang sedang dilaksanakan beserta nilai yang sedang diperiksa. Jika elemen yang dicari ditemukan, fungsi mengembalikan indeks elemen tersebut dan jumlah iterasi yang dilakukan. Jika tidak ditemukan, fungsi mengembalikan -1 dan jumlah iterasi. Selanjutnya Fungsi binarySearch juga menerima array dan target yang sama, tetapi sebelum memanggil fungsi ini array harus diurutkan. Pencarian biner melakukan proses dengan membagi array menjadi dua bagian dan memeriksa elemen tengah. Apabila elemen tengah tersebut sama dengan target yang dicari, indeksnya akan dikembalikan. Jika target ternyata lebih kecil dari elemen tengah maka pencarian akan dilanjutkan di bagian kiri sementara jika target lebih besar pencarian akan berlanjut di bagian kanan. Sama seperti pada pencarian sekuensial setiap langkah dalam proses ini juga dicetak untuk menunjukkan perkembangan pencarian. Pada fungsi main program pertama-tama mendefinisikan sebuah array yang tidak terurut serta target yang ingin dicari. Program kemudian melakukan pencarian sekuensial dan menampilkan hasilnya. Setelah array diurutkan dengan menggunakan sort. Float64s, pencarian biner dilakukan pada array yang telah terurut. Hasil dari kedua metode pencarian, baik yang berhasil maupun yang tidak, dicetak ke layar, termasuk jumlah langkah yang diperlukan untuk menemukan target.

2. Guided2

Source code:

```
//Feros Pedrosa

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:

Program diatas ditulis dalam bahasa go dan berfungsi untuk melakukan pencarian data mahasiswa menggunakan dua metode pencarian yaitu sekuensial dan biner. Program ini mendefinisikan sebuah struktur data yang disebut mahasiswa, yang menyimpan informasi terkait mahasiswa, seperti nama, NIM, kelas, jurusan, dan IPK. Struktur ini selanjutnya digunakan untuk membuat array bernama arrMhs, yang dapat menampung hingga 2023 mahasiswa. Fungsi SeqSearch_3 diimplementasikan untuk melakukan pencarian secara berurutan berdasarkan nama mahasiswa. Fungsi ini mengambil tiga parameter: array mahasiswa, jumlah elemen yang akan dicari, dan nama yang ingin dicari. Apabila nama yang dicari ditemukan, fungsi ini akan mengembalikan indeks dari nama tersebut beserta jumlah iterasi yang dilakukan selama proses pencarian. Namun, jika nama tersebut tidak ditemukan, fungsi akan mengembalikan -1

sebagai indikasi bahwa pencarian tidak berhasil. Di sisi lain, fungsi BinarySearch_3 berfungsi untuk mencari mahasiswa berdasarkan NIM. Namun, sebelum melakukan pencarian, data mahasiswa harus diurutkan terlebih dahulu berdasarkan NIM. Fungsi ini menerima tiga parameter: array mahasiswa, jumlah elemen, dan NIM yang dicari. Jika NIM ditemukan, fungsi ini akan mengembalikan indeks serta jumlah iterasi yang dilakukan. Sebaliknya, jika tidak ditemukan, fungsi akan mengembalikan nilai -1. Dalam fungsi main, program ini melakukan pengisian data mahasiswa secara manual dengan total sepuluh entri. Selanjutnya, program menggunakan metode Pencarian Berurutan (Sequential Search) untuk mencari nama “Fajar” dan mencetak hasil pencariannya, termasuk indeks serta jumlah iterasi yang dilakukan. Setelah itu, data akan diurutkan berdasarkan NIM dengan menggunakan fungsi sort. Slice. Kemudian, program melanjutkan dengan melakukan Pencarian Biner (Binary Search) untuk mencari NIM “2206”. Hasil dari pencarian ini akan dicetak, menunjukkan apakah NIM tersebut ditemukan, beserta indeks dan jumlah iterasi yang dilakukan.

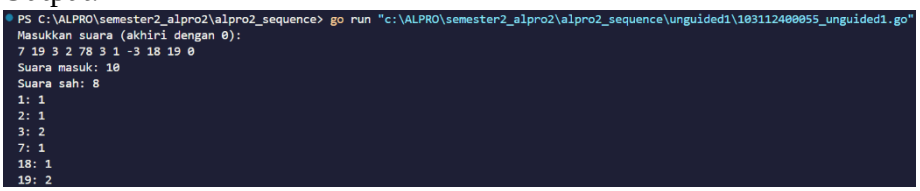
III. UNGUIDED

1. Unguided1

Source code:

```
//Feros Pedrosa

package main

import "fmt"

func main() {
   var input int
   votes := make([]int, 21)

   fmt.Println("Masukkan suara (akhiri dengan 0):")
   count := 0
   validVotes := 0

   for {
      fmt.Scan(&input)
      if input == 0 {
         break
      }
      if input >= 1 && input <= 20 {
         votes[input]++
         validVotes++
      }

      count++
   }

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

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

Output:



Deskripsi program:

Program diatas ditulis dalam bahasa go dan berfungsi menghitung dan mencatat suara yang masuk dalam pemilihan. Pertama deklarasikan variabel input untuk menyimpan suara yang dimasukkan oleh pengguna serta array votes dengan panjang 21 untuk menyimpan jumlah suara untuk setiap kandidat di mana indeks 1 hingga 20 mewakili kandidat yang berbeda. Setelah itu, program meminta pengguna untuk memberikan suara dengan petunjuk bahwa proses dapat dihentikan dengan memasukkan angka 0. Program kemudian memasuki sebuah loop tanpa batas yang akan terus meminta input dari pengguna. Jika pengguna memasukkan 0, loop tersebut akan berakhir. Jika input yang dimasukkan berada dalam kisaran 1 sampai 20 program akan menambahkan suara untuk kandidat yang sesuai dan juga menghitung jumlah suara yang sah. Variabel count digunakan untuk mencatat total suara yang telah dimasukkan, termasuk suara yang tidak valid. Setelah pengguna selesai memasukkan suara, program mencetak total suara yang masuk dan jumlah suara yang sah. Terakhir, program menampilkan jumlah suara untuk setiap kandidat yang menerima suara, hanya jika jumlah suara untuk kandidat tersebut lebih dari nol.

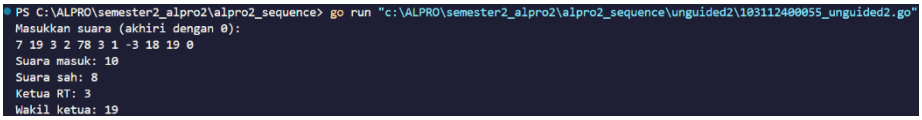
2. Unguided2

Source code:

```
//Feros Pedrosa

package main

import "fmt"

func main() {
   var input int
   votes := make([]int, 21) // Indeks 1-20 untuk nomor calon

   fmt.Println("Masukkan suara (akhiri dengan 0):")
   count := 0
   validVotes := 0

   for {
      fmt.Scan(&input)
      if input == 0 {
         break
      }
      count++
      if input >= 1 && input <= 20 {
         votes[input]++
         validVotes++
      }
   }

   ketua, suaraKetua := 0, 0
   wakil, suaraWakil := 0, 0

   for i := 1; i <= 20; i++ {
      suaraSaatIni := votes[i]
      if suaraSaatIni == 0 {
         continue
      }

      if suaraSaatIni > suaraKetua {
         wakil, suaraWakil = ketua, suaraKetua
         ketua, suaraKetua = i, suaraSaatIni
      } else if suaraSaatIni == suaraKetua {
         if i < ketua {
            wakil, suaraWakil = ketua, suaraKetua
            ketua, suaraKetua = i, suaraSaatIni
         } else if suaraWakil < suaraSaatIni || (suaraWakil ==
suaraSaatIni && i < wakil) {
```

```
                wakil, suaraWakil = i, suaraSaatIni
            }
        } else if suaraSaatIni > suaraWakil {
            wakil, suaraWakil = i, suaraSaatIni
        } else if suaraSaatIni == suaraWakil && i < wakil {
            wakil, suaraWakil = i, suaraSaatIni
        }
    }

    fmt.Printf("Suara masuk: %d\n", count)
    fmt.Printf("Suara sah: %d\n", validVotes)
    fmt.Printf("Ketua RT: %d\n", ketua)
    fmt.Printf("Wakil ketua: %d\n", wakil)
}
```

Output:



Deskripsi program:

Program diatas ditulis dalam bahasa go dan berfungsi menghitung suara yang diberikan kepada calon ketua dan wakil ketua, dengan batasan bahwa hanya ada 20 calon yang dapat dipilih. Program dimulai dengan deklarasi variabel input yang berguna untuk menyimpan suara dari pengguna, serta array votes yang berfungsi untuk mencatat jumlah suara untuk setiap calon, mulai dari indeks 1 hingga 20. Pengguna diminta untuk memasukkan suara mereka, dan proses pemungutan suara akan berakhir ketika pengguna memasukkan angka 0. Selama proses ini, program juga menghitung total suara yang diterima serta jumlah suara yang valid, yaitu suara yang berada dalam rentang 1 hingga 20. Setelah pemungutan suara selesai, program akan melanjutkan untuk menentukan calon ketua dan wakil ketua berdasarkan jumlah suara yang didapat. Dua variabel, yaitu ketua dan wakil, serta suaraKetua dan suaraWakil, digunakan untuk menyimpan nomor calon dan jumlah suara yang diperoleh. Program akan melakukan iterasi pada array votes untuk menemukan calon dengan jumlah suara terbanyak. Jika ada calon yang memiliki jumlah suara yang sama, maka program akan memilih calon dengan nomor yang lebih kecil untuk dijadikan ketua atau wakil ketua. Terakhir, hasil pemungutan suara akan ditampilkan, termasuk total suara yang diterima, jumlah suara yang sah, serta nomor calon yang terpilih sebagai ketua dan wakil ketua.

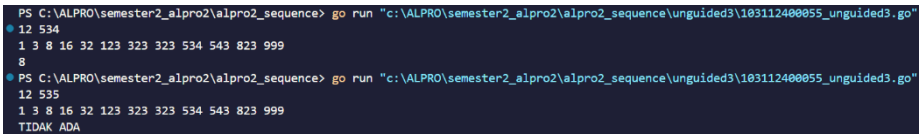
3. Unguided3

Source code:

```
//Feros Pedrosa

package main

import "fmt"

const NMAX = 1000000

var data [NMAX]int

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

    isiArray(n)

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

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

func posisi(n, k int) int {
    low := 0
    high := n - 1

    for low <= high {
        mid := (low + high) / 2
        if data[mid] == k {
            return mid
        } else if data[mid] < k {
            low = mid + 1
        } else {
            high = mid - 1
        }
    }
```

```
        return -1
    }
```

Output:



Deskripsi program:

Program diatas ditulis dalam bahasa go dan berfungsi mencari posisi suatu bilangan k dalam sebuah array data yang berisi n bilangan bulat positif yang telah terurut secara membesar. Proses pencarian dilakukan dengan menggunakan metode pencarian biner yang efektif. Program ini dimulai dengan mendeklarasikan konstanta NMAX sebagai batas maksimum untuk elemen array yaitu satu juta, serta mendefinisikan array global bernama data untuk menyimpan masukan angka. Fungsi main memulai jalannya program dengan membaca dua bilangan bulat: n (jumlah data) dan k (bilangan yang ingin dicari). Selanjutnya fungsi isiArray dipanggil untuk mengisi array data dengan n angka bulat yang diambil dari input. Pengisian array dilakukan melalui perulangan for yang dilakukan sebanyak n kali di mana setiap angka yang dibaca disimpan ke dalam array data. Selanjutnya fungsi posisi dipanggil untuk menemukan letak angka k dalam array. Fungsi ini menerapkan algoritma pencarian biner yang sangat efisien untuk data yang telah diurutkan. Selama proses ini, fungsi menggunakan dua indeks, yaitu low dan high, yang menandakan batas pencarian. Indeks tengah (mid) dihitung dan dibandingkan dengan nilai k. Jika elemen tengah sama dengan k, maka indeks tersebut dikembalikan sebagai hasil pencarian. Jika k lebih besar daripada elemen tengah, pencarian dilanjutkan pada bagian kanan array. Di sisi lain, jika k lebih kecil pencarian berlanjut pada bagian kiri. Proses ini terus berlanjut hingga low melebihi high, yang menunjukkan bahwa elemen k tidak ada dalam array. Jika fungsi posisi mengembalikan nilai -1, maka program akan mencetak “TIDAK ADA”. Namun jika k ditemukan, maka indeks dari elemen tersebut dicetak. Output indeks dimulai dari 0 sesuai dengan standar indeks array di Go.

IV. KESIMPULAN

Praktikum ini membahas tentang pencarian nilai dalam himpunan data dengan fokus pada dua algoritma utama yaitu Sequential Search dan Binary Search. Pencarian nilai acak pada himpunan data sering kali diperlukan untuk menemukan data spesifik, seperti nama mahasiswa atau alamat rumah. Dalam Sequential Search, data diperiksa satu per satu dari awal hingga akhir dan proses pencarian berhenti saat data yang dicari ditemukan atau semua data telah diperiksa. Algoritma ini sederhana namun kurang efisien untuk dataset besar. Sebaliknya, Binary Search memerlukan data yang terurut dan bekerja dengan membagi rentang pencarian menjadi dua, sehingga lebih efisien dibandingkan Sequential Search terutama untuk dataset besar. Praktikum ini juga menjelaskan bagaimana algoritma pencarian dapat diterapkan pada array bertipe data struct.

V. REFERENSI

MODUL 11. PENCARIAN NILAI ACAK PADA HIMPUNAN DATA - Praktikum Alpro 2