ALGORITMA PEMOGRAMAN 2

MODUL 8

"PENCARIAN NILA ACAK PADA HIMPUNAN DATA”

Oleh:

NAMA: M. DAVI ILYAS RENALDO

NIM: 103112400062

KELAS: 12-IF-01

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

I.DASAR TEORI

11.1 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.

Berikut ini adalah contoh notasi algoritma sekuensial search sesuai dengan penjelasan di atas.

	Notasi Algoritma	Notasi dalam bahasa Go
1	found ← false	found = false
2	i ← 0	i = 0
3	while i < n and not found do	for i < n && !found {
4	found ← T[i] == X	found = T[i] == X
5	i ← i + 1	i = i + 1
6	endwhile	}

Adapun contoh potongan program pencarian sebuah string pada array of string adalah sebagai berikut ini:

```
..  ...
5   type arrStr [1234]string
..  ...
15
16  func SeqSearch_1(T arrStr, n int, X string) bool {
17  /* mengembalikan true apabila X ditemukan di dalam array T yang berisi n buah
18  teks, atau false apabila X tidak ditemukan */
19      var found bool = false
20      var j int = 0
21      for j < n && !found {
22          found = T[j] == X
23          j = j + 1
24      }
25      return found
26  }
```

Seperti penjelasan yang sudah diberikan sebelumnya, bahwa pada pencarian indeks atau lokasi dari data yang dicari lebih penting dibandingkan data itu sendiri. Oleh karena itu algoritma pencarian di atas bisa dimodifikasi untuk menghasilkan indeks dari data yang dicari. Selain itu perlu dipersiapkan juga sebuah nilai untuk menyatakan apabila pencarian data tidak ditemukan, biasanya -1 atau bilangan di luar indeks dari array yang valid.

```
..  ...
5   type arrStr [1234]string
..  ...
15
16  func SeqSearch_1(T arrStr, n int, X string) int {
17  /* mengembalikan indeks dari X apabila X ditemukan di dalam array T yang berisi
18  n buah teks, atau -1 apabila X tidak ditemukan */
19      var found int = -1
20      var j int = 0
21      for j < n && found == -1 {
22          if T[j] == X {
23              found = j
24          }
25          j = j + 1
26      }
27      return found
28  }
```

Variasi yang lain

```
..  ...
5   type arrStr [1234]string
..  ...
15
16  func SeqSearch_1(T arrStr, n int, X string) int {
17
```

```
18  /* mengembalikan indeks dari X apabila X ditemukan di dalam array T yang berisi
19  n buah teks, atau -1 apabila X tidak ditemukan */
20      var j int = 0
21      for j < n-1 && T[j] != X {
22          j = j + 1
23      }
24      if T[j] == X {
25          return j
26      }else{
27          return -1
28      }
29  }
```

11.2 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.

Algoritma ini dikenal dengan nama **Binary Search**. Berikut ini adalah contoh notasi algoritma binary search sesuai dengan penjelasan di atas.

	Notasi Algoritma	Notasi dalam bahasa Go
1	kr ← 0	kr = 0
2	kn ← n-1	kn = n-1
3	found ← false	found = false
4	while kr <= kn and not found do	for kr <= kn && !found {
5	med ← (kr+kn) div 2	med = (kr+kn) / 2
6	if a[med] < X then	if a[med] < X {
7	kr ← med + 1	kr = med + 1
8	else if a[med] > X then	} else if a[med] > X {
9	kn ← med − 1	kn = med
10	else	} else {
11	found ← true	found = true
12	endif	}
13	endwhile	}

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. Selanjutnya bagaimana contoh algoritma pencarian data apabila dituliskan ke dalam suatu fungsi pencarian dan diketahui array terurut secara mengecil atau descending.

```
..  ...
5   type arrInt [4321]int
..  ...
15  func BinarySearch_1(T arrInt, n int, X string) bool {
16  /* mengembalikan true apabila X ditemukan di dalam array T yang berisi n buah
17  bilangan bulat terurut secara descending/mengecil, atau false apabila X tidak
18  ditemukan */
19      var found bool = false
20      var med int
21      var kr int = 0
22      var kn int = n - 1
23      for kr <= kn && !found {
24          med = (kr + kn) / 2
25          if X > T[med] {
26              kn = med - 1
27          }else if X < T[med] {
28              kr = med + 1
29          }else{
30              found = true
31          }
32      }
33      return found
34  }
35
```

Adapun variasi lain yang mengembalikan indeks dari data yang dicari adalah sebagai berikut:

```
..  ...
5   type arrInt [4321]int
..  ...
15  func BinarySearch_2(T arrInt, n int, X string) int {
16  /* mengembalikan indeks dari X apabila X ditemukan di dalam array T yang berisi
17  n buah bilangan bulat terurut secara descending/mengecil, atau -1 apabila X
18  tidak ditemukan */
19      var found int = -1
20      var med int
21      var kr int = 0
22      var kn int = n - 1
23      for kr <= kn && found == -1 {
24          med = (kr + kn) / 2
25          if X > T[med] {
26              kn = med - 1
27          }else if X < T[med] {
28              kr = med + 1
29          }else{
30              found = med
31          }
32      }
33      return found
34  }
```

Proses binary search akan berakhir apabila nilai kr > kn (data tidak ditemukan) atau found sudah tidak bernilai false atau -1 (data ditemukan).

11.3 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.

Berikut adalah contoh apabila array mahasiswa terurut berdasarkan nim secara membesar (ascending) dan dilakukan pencarian menggunakan algoritma sekuensial dan biner.

```
...
type mahasiswa struct {
    nama, nim, kelas, jurusan string
    ipk float64
}
type arrMhs [2023]mahasiswa
...
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
}

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
}
```

II. 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("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 arr[mid] < target {
            low = mid + 1
        } else {
            high = 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 := sequentialSearch(data, target)
    if idxSeq != -1 {
```

```
        fmt.Printf("Hasil: Ditemukan pada index %d dalam %d langkah\n\n", idxSeq, iterSeq)
    } else {
        fmt.Printf("Hasil: Tidak ditemukan setelah %d langkah\n", iterSeq)
    }

    sort.Float64s(data)
    fmt.Println("\nBinary Search (setelah data di urutkan):")
    fmt.Println("Data urut:", data)

    idxBin, iterBin := binarySearch(data, target)
    if idxBin != -1 {
        fmt.Printf("Hasil: Ditemukan pada index %d dalam %d langkah\n", idxBin, iterBin)
    } else {
        fmt.Printf("Hasil: Tidak ditemukan setelah %d langkah\n", iterBin)
    }

}
```

OUTPUT:

```
PS C:\Users\ACER\OneDrive\alpro2\modul1> go run "c:\Users\ACER\OneDrive\alpro2\modul1\GUIDED 8\1.go"
Sequential Search (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 pada index 7 dalam 8 langkah


Binary Search (setelah data di urutkan):
Data urut: [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
Hasil: Ditemukan pada index 6 dalam 4 langkah
PS C:\Users\ACER\OneDrive\alpro2\modul1>
```

DESKRIPSI:

Program ini adalah program yang mencari angka 13.0 di dalam array dengan dua cara (berurutan dan biner) kemudian mencetak langkah-langkah pencarian, lalu program ini menampilkan hasil indeks dan jumlah langkah untuk membandingkan efisiensi kedua metode.

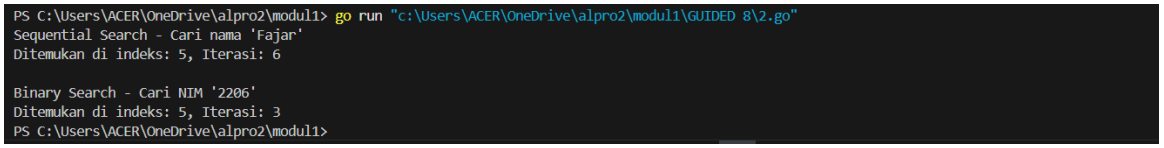
2.

```
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 ini adalah program yang berfungsi mencari data mahasiswa dengan dua metode (Sequential Search dan Binary Search). Kemudian program ini akan berjalan mencari mahasiswa berdasarkan nama dan NIM.

III. UNGUIDED

1.

```
// M. DAVI ILYAS RENALDO_103112400062
package main
import (
    "fmt"
)

func main() {
    var suara [21]int
    var angka int
    suaraMasuk := 0
    suaraSah := 0

    for {
        _, err := fmt.Scan(&angka)
        if err != nil || angka == 0 {
            break
        }

        suaraMasuk++

        if angka >= 1 && angka <= 20 {
            suara[angka]++
            suaraSah++
        }
    }
    fmt.Printf("Suara masuk: %d\n", suaraMasuk)
    fmt.Printf("Suara sah: %d\n", suaraSah)

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

OUTPUT:

```
PS C:\Users\ACER\OneDrive\alpro2\modul1> go run "c:\Users\ACER\OneDrive\alpro2\modul1\UNGUIDED 8\tempCodeRunnerFile.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 C:\Users\ACER\OneDrive\alpro2\modul1>
```

DESKRIPSI:

Program ini adalah program untuk menghitung jumlah suara yang masuk dan menentukan suara sah berdasarkan input angka dari pengguna, suara dianggap sah jika bernilai antara 1 hingga 20, kemudian program ini akan mencatat jumlah suara untuk setiap kandidat serta menampilkan hasilnya setelah input selesai.

2.

```
// M. DAVI ILYAS RENALDO_103112400062
package main
import (
    "fmt"
    "sort"
)

type Pilkart struct {
    totalSuara   int
    suaraSah     int
    hasilPilih map[int]int
}

func prosesSuara(data []int) (Pilkart, int) {
    p := Pilkart{
        totalSuara:   0,
        suaraSah:     0,
        hasilPilih: make(map[int]int),
    }

    for i := 1; i <= 20; i++ {
        p.hasilPilih[i] = 0
    }

    iterasi := 0

    for _, suara := range data {
        iterasi++
        p.totalSuara++

        if suara == 0 {
            p.totalSuara--
            break
        }

        if suara >= 1 && suara <= 20 {
            p.suaraSah++
            p.hasilPilih[suara]++
        }
    }

    return p, iterasi
}

func tentukanPemenang(p Pilkart) (int, int) {
    type Calon struct {
        nomor int
        suara int
    }

    var noCalon []Calon

    for nomor, suara := range p.hasilPilih {
        if suara > 0 {
            noCalon = append(noCalon, Calon{nomor, suara})
```

```
        }
    }

    sort.Slice(noCalon, func(i, j int) bool {
        if noCalon[i].suara == noCalon[j].suara {
            return noCalon[i].nomor < noCalon[j].nomor
        }
        return noCalon[i].suara > noCalon[j].suara
    })

    var ketua, wakil int

    if len(noCalon) > 0 {
        ketua = noCalon[0].nomor
    }

    if len(noCalon) > 1 {
        wakil = noCalon[1].nomor
    }

    return ketua, wakil
}

func main() {
    var input int
    var data []int

    for {
        fmt.Scan(&input)
        data = append(data, input)
        if input == 0 {
            break
        }
    }

    hasilPungutan, _ := prosesSuara(data)

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

    ketua, wakil := tentukanPemenang(hasilPungutan)
    fmt.Printf("Ketua RT: %d\n", ketua)
    fmt.Printf("Wakil ketua: %d\n", wakil)
}
```

OUTPUT:

```
PS C:\Users\ACER\OneDrive\alpro2\modul1> go run "c:\Users\ACER\OneDrive\alpro2\modul1\UNGUIDED 8\2.go"
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 ini adalah program yang mensimulasikan proses pemungutan suara pemilihin ketua dan wakil RT.pengguna diminta untuk memasukkan jumlah suara setiap kandidat dan program ini akan berhenti ketika angka yang dimasukkan 0. Kemudian program akan berjalan dan mencetak hasilnya .

3.

```
// M. DAVI ILYAS RENALDO_103112400062
package main
import "fmt"

const NMAX = 1000000
var data [NMAX]int

func main() {
    var n, k int

    fmt.Scan(&n, &k)

    isiArray(n)

    index := posisi(n, k)

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

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
    var tengah int

    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:

```
PS C:\Users\ACER\OneDrive\alpro2\modul1> go run "c:\Users\ACER\OneDrive\alpro2\modul1\UNGUIDED 8\3.go"
12 534
1 3 8 16 32 123 323 323 534 543 823 999
8
PS C:\Users\ACER\OneDrive\alpro2\modul1> go run "c:\Users\ACER\OneDrive\alpro2\modul1\UNGUIDED 8\3.go"
12 535
1 3 8 16 32 123 323 323 534 543 823 999
TIDAK ADA
PS C:\Users\ACER\OneDrive\alpro2\modul1>
```

DESKRIPSI:
Program ini adalah program yang membaca sebuah bilangan n sebagai jumlah elemen dan bilangan k sebagai nilai yang dicari, kemudian mengisi array data sebanyak n elemen dari input. Setelah itu, program menggunakan algoritma binary search melalui fungsi posisi untuk mencari indeks dari nilai k dalam array yang diasumsikan sudah terurut secara menaik. Jika nilai k ditemukan maka program mencetak indeksnya dan jika tidak maka program mencetak "TIDAK ADA".

IV.KESIMPULAN

V.REFERENSI

- modul 11. Pencarian nilai acak pada himpunan data. Praktikum alpro2