

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2
MODUL 11
PENCARIAN NILAI ACAK PADA HIMPUNAN DATA**



Oleh:

NAMA: Lutfi Shidqi Mardian

NIM: 103112400077

KELAS: IF-12-01

**S1 TEKNIK INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO**

2025

I. DASAR TEORI

Tipe Bentuk (Struct dan Alias)

Dalam bahasa pemrograman Go (*Golang*), dikenal berbagai jenis **tipe bentuk** yang memungkinkan programmer untuk membuat struktur data kompleks dan lebih sesuai dengan kebutuhan aplikasi. Beberapa tipe bentuk yang sering digunakan dalam Go antara lain: **alias, struct, array, slice, dan map**. Berikut penjelasan masing-masing:

- **Alias (Type)**

Alias adalah pemberian nama baru untuk tipe data yang sudah ada, sehingga lebih mudah dibaca atau digunakan. Dalam Go, alias didefinisikan menggunakan kata kunci `type`.

Contoh:

```
type bilangan int  
  
type pecahan float64
```

Dengan cara ini, tipe `int` dan `float64` bisa diakses menggunakan nama baru yang lebih spesifik.

- **Struct (Structure/Record)**

Struct adalah kumpulan beberapa variabel yang memiliki hubungan, digabung menjadi satu kesatuan. Setiap elemen di dalam struct disebut `field`.

Contoh deklarasi struct di Go:

```
type waktu struct {  
    jam, menit, detik int  
}
```

Struct berguna untuk mengelompokkan data yang berkaitan, seperti data waktu, koordinat, atau informasi lainnya.

- **Array**

Array adalah kumpulan elemen dengan tipe data yang sama dan jumlah elemen yang tetap (statis) selama program berjalan. Di Go, deklarasi array menentukan jumlah elemen secara eksplisit.

Contoh deklarasi array:

```
var arr [5]int  
var buf = [5]byte{7, 3, 5, 2, 11}
```

Beberapa hal penting tentang array:

1. Indeks array di Go dimulai dari 0.
2. Ukuran array bisa diperiksa menggunakan fungsi `len(array)`.

3. Elemen dapat diakses dan dimodifikasi menggunakan indeks, misalnya `arr[0] = 10`.

- **Slice**

Slice adalah tipe data di Go yang mirip array tetapi memiliki ukuran yang bisa berubah selama eksekusi program. Slice lebih fleksibel dibandingkan array biasa. Slice bisa dibuat dari array, slice lain, atau menggunakan fungsi `make`.

Contoh deklarasi slice:

```
var s1 = []int{1, 2, 3, 4}

var s2 = make([]int, 10, 20)
```

Beberapa fungsi yang umum digunakan pada slice:

1. `len(slice)`: Mengembalikan jumlah elemen dalam slice.
2. `cap(slice)`: Mengembalikan kapasitas maksimum slice.
3. `append(slice, elemen)`: Menambahkan elemen baru ke slice

mfjmfj

- **Map**

Map adalah tipe data asosiatif di Go yang menyimpan pasangan key-value. Tidak seperti array, indeks pada map bisa berupa tipe data apapun (string, integer, float, dll).

Contoh deklarasi map:

```
var dct map[string]int

dct = map[string]int{"john": 25, "anne": 30}
```

II. GUIDED

1.

```
package main

import (
    "fmt"
    "sort"
)

func sequentialSearch(arr []float64, target float64) (int, int) {
    itrs := 0
    for i, val := range arr {
        itrs++
        fmt.Printf("Sequential Step %d: cek arr[%d] = %.1f\n",
            itrs, i, val)
        if val == target {
            return i, itrs
        }
    }
    return -1, itrs
}

func binarySeacrh(arr []float64, target float64) (int, int) {
    itrs := 0
    low := 0
    high := len(arr) - 1
    for low <= high {
        itrs++
        mid := (low + high) / 2
        fmt.Printf("Binary Step %d: cek arr[%d] = %.1f\n", itrs,
            mid, arr[mid])
    }
}
```

```

        if arr[mid] == target {
            return mid, itr
        } else if target < arr[mid] {
            high = mid - 1
        } else {
            low = mid + 1
        }
    }
    return -1, itr
}

func main() {
    //sequential
    data := []float64{2, 7, 9, 1, 5, 6, 18, 13, 25, 20}
    fmt.Printf("Len(data)\n")
    target := 13.0

    fmt.Println("Sequential Search (data tidak perluurut): ")
    idxSeq, iterSeq := sequentialSearch(data, target)
    if idxSeq != -1 {
        fmt.Printf("Hasil: Ditemukan di indeks %d dalam %d Langkah\n", idxSeq, iterSeq)
    } else {
        fmt.Printf("Hasil: Tidak ditemukan setelah %d Langkah\n", iterSeq)
    }

    //Binary
    sort.Float64s(data)
    fmt.Println("Binary Search (setelah data diurutkan): ")

```

```
fmt.Println("Data terurut: ", data)

idxBin, itrBin := binarySeacrh(data, target)

if idxBin != -1 {

    fmt.Printf("Hasil: Ditemukan di indeks %d dalam %d
Langkah\n", idxBin, itrBin)

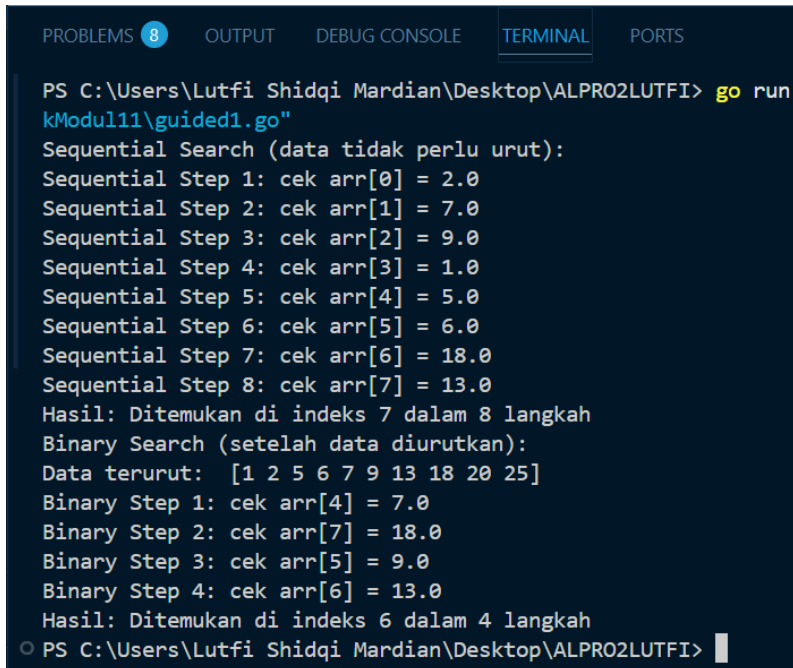
} else {

    fmt.Printf("Hasil: Tidak ditemukan setelah %d Langkah\n",
itrBin)

}

}
```

Output Screenshot:



```
PROBLEMS 8 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI> go run
kModul11\guided1.go"
Sequential Search (data tidak perluurut):
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
Hasil: Ditemukan di indeks 6 dalam 4 langkah
PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI>
```

Penjelasan:

Program ini mengimplementasikan dua metode pencarian nilai dalam array bertipe float64, yaitu **sequential search** dan **binary search**. Pertama, program mencari nilai target (13.0) secara berurutan dalam array acak tanpa perlu pengurutan, lalu mencetak langkah-langkah pencariannya. Setelah itu, array diurutkan menggunakan `sort.Float64s`, dan dilakukan pencarian biner yang lebih efisien dengan mencetak setiap langkah pemeriksaannya. Program menampilkan apakah nilai ditemukan beserta indeks dan jumlah langkah pencarian untuk masing-masing 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 Screenshot:

```
PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI> go run kModul11\guided2.go"
Sequential Search - Cari nama 'Fajar'
Ditemukan di indeks: 5, Iterasi: 6

Binary Search - Cari NIM '2206'
Ditemukan di indeks: 5, Iterasi: 3
PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI> 
```

Penjelasan:

Program ini mendemonstrasikan dua metode pencarian data pada array bertipe struct mahasiswa, yaitu **Sequential Search** berdasarkan *nama* dan **Binary Search** berdasarkan *NIM*. Struct mahasiswa menyimpan informasi seperti nama, NIM, kelas, jurusan, dan IPK. Pertama, program melakukan pencarian nama mahasiswa menggunakan sequential search tanpa perlu mengurutkan data. Selanjutnya, program mengurutkan data mahasiswa berdasarkan NIM menggunakan sort.Slice, lalu melakukan binary search untuk mencari NIM tertentu dengan efisiensi yang lebih tinggi. Output program menunjukkan hasil pencarian beserta indeks dan jumlah iterasi yang dilakukan untuk menemukan data.

III. UNGUIDED

1.

```
package main

import "fmt"

func main() {

    data := []int{7, 19, 3, 2, 78, 3, 1, -3, 18, 19, 0}

    input := 0

    valid := 0

    var hitungSuara [21]int

    for _, suara := range data {

        if suara == 0 {

            break

        }

        input++

        if suara >= 1 && suara <= 20 {

            valid++

            hitungSuara[suara]++

        }

    }

    fmt.Printf("Suara masuk: %d\n", input)

    fmt.Printf("Suara sah: %d\n", valid)

    for calon := 1; calon <= 20; calon++ {

        if hitungSuara[calon] > 0 {

            fmt.Printf("%d: %d\n", calon, hitungSuara[calon])

        }

    }

}
```

Output Screenshot:

A screenshot of a Go IDE's terminal window. The terminal has tabs for PROBLEMS (8), OUTPUT, DEBUG CONSOLE, TERMINAL (selected), and PORTS. The command prompt shows the user running 'go run kModul11\unguided1.go'. The program outputs 'Suara masuk: 10' and 'Suara sah: 8', followed by a list of votes for candidates 1 through 19: '1: 1', '2: 1', '3: 2', '7: 1', '18: 1', and '19: 2'. The prompt then returns to the shell.

```
PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI> go run kModul11\unguided1.go
Suara masuk: 10
Suara sah: 8
1: 1
2: 1
3: 2
7: 1
18: 1
19: 2
PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI>
```

Penjelasan:

Program ini menghitung hasil pemungutan suara dari data yang berisi angka-angka representasi calon, dengan rentang calon nomor 1 hingga 20. Setiap elemen array data dianggap sebagai satu suara. Proses berhenti jika ditemukan angka 0 dalam data. Program mencatat jumlah seluruh suara yang masuk (input), jumlah suara yang sah (hanya yang bernilai antara 1–20), serta mencatat berapa kali setiap calon memperoleh suara menggunakan array `hitungSuara`. Di akhir, program menampilkan jumlah suara masuk, suara sah, dan jumlah suara yang diperoleh masing-masing calon yang mendapat minimal satu suara.

2.

```
package main

import "fmt"

func main() {

    data := []int{7, 19, 3, 2, 78, 3, -1, -3, 18, 19, 0}

    input := 0

    valid := 0

    var hitungSuara [21]int

    for _, suara := range data {

        if suara == 0 {

            break

        }

        input++

        if suara >= 1 && suara <= 20 {

            valid++

            hitungSuara[suara]++

        }

    }

    max1, max2 := 0, 0

    ketua, wakil := 0, 0

    for i := 1; i <= 20; i++ {

        jumlah := hitungSuara[i]

        if jumlah > max1 {

            max2 = max1

            wakil = ketua

            max1 = jumlah

            ketua = i

        }

    }

    fmt.Println("Ketua:", ketua, "Wakil:", wakil)

}
```

```

        } else if jumlah == max1 && i < ketua {

            max2 = max1

            wakil = ketua

            ketua = i

        } else if jumlah > max2 && jumlah < max1 {

            max2 = jumlah

            wakil = i

        } else if jumlah == max2 && jumlah != 0 && i < wakil {

            wakil = i

        }

    }

    fmt.Printf("Suara masuk: %d\n", input)

    fmt.Printf("Suara sah: %d\n", valid)

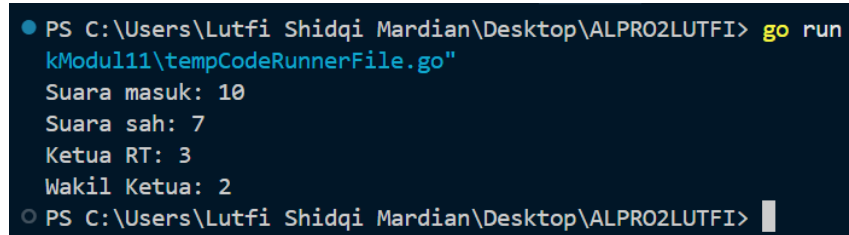
    fmt.Printf("Ketua RT: %d\n", ketua)

    fmt.Printf("Wakil Ketua: %d\n", wakil)

}

```


Output Screenshot:



```
PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI> go run kModul11\tempCodeRunnerFile.go
Suara masuk: 10
Suara sah: 7
Ketua RT: 3
Wakil Ketua: 2
PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI>
```

Penjelasan:

Program ini memproses data hasil pemungutan suara untuk memilih Ketua dan Wakil Ketua RT dari calon bernomor 1 hingga 20. Suara disimpan dalam slice data, dan proses berhenti saat menemukan angka 0. Program mencatat jumlah suara masuk (input) dan jumlah suara sah (valid), lalu menghitung suara sah setiap calon menggunakan array `hitungSuara`. Setelah itu, program menentukan calon dengan suara terbanyak sebagai Ketua RT dan calon dengan suara terbanyak kedua sebagai Wakil Ketua RT, dengan prioritas pada nomor calon lebih kecil jika ada jumlah suara yang sama. Hasil akhirnya adalah informasi jumlah suara, serta nomor calon Ketua dan Wakil Ketua RT.

3.

```
package main

import "fmt"

const MAX = 100000

var data [MAX]int

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

func posisi(k, n int) int {
    for i := 0; i < n; i++ {
        if data[i] == k {
            return i
        }
    }
    return -1
}

func main() {
    var n, k int

    fmt.Scan(&n, &k)

    isiArray(n)

    pos := posisi(k, n)

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

Output Screenshot:

```
● PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI> go run
  kModul11\unguided3.go"
    12 534
    1 3 8 16 32 123 323 323 534 543 823 999
    8
● PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI> go run
  kModul11\unguided3.go"
    12 535
    1 3 8 16 32 123 323 323 534 543 823 999
    TIDAK ADA
○ PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI> █
```

Penjelasan:

Program ini melakukan pencarian posisi elemen tertentu dalam sebuah array. Pertama, program membaca dua input: n (jumlah elemen yang akan diisi ke array) dan k (nilai yang ingin dicari). Kemudian, fungsi isiArray digunakan untuk mengisi array data sebanyak n elemen dari input pengguna. Setelah itu, fungsi posisi melakukan pencarian nilai k secara sekuensial dalam array. Jika ditemukan, program mencetak indeks (posisi) dari elemen tersebut; jika tidak ditemukan, program mencetak "TIDAK ADA". Program ini efisien untuk input besar karena mendukung hingga 100.000 elemen.

IV. KESIMPULAN

Berdasarkan hasil praktikum dan implementasi dari algoritma pencarian, dapat disimpulkan bahwa algoritma *sequential search* dan *binary search* memiliki karakteristik serta kelebihan masing-masing dalam proses pencarian data. *Sequential search* sangat cocok digunakan pada data yang tidak terurut karena pencarian dilakukan satu per satu dari awal hingga akhir array. Meskipun sederhana, algoritma ini tidak efisien untuk dataset yang besar. Sebaliknya, *binary search* jauh lebih efisien karena menggunakan prinsip pembagian dua, namun memerlukan data yang telah terurut sesuai kriteria pencarian. Dalam konteks praktikum ini, penggunaan kedua metode ini membantu memahami pentingnya pemilihan algoritma yang tepat berdasarkan struktur data dan kondisi dataset. Pemahaman algoritma pencarian ini sangat penting dalam pengembangan program yang efisien, terutama saat bekerja dengan kumpulan data yang besar dan kompleks.

V. REFERENSI

1. **Tim Dosen Informatika Telkom University Purwokerto.** (2024). *Modul Praktikum Algoritma dan Pemrograman 2 – Modul 8: Pencarian Nilai Acak pada Himpunan Data*.
2. Donovan, A. A., & Kernighan, B. W. (2015). *The Go Programming Language*. Addison-Wesley.
3. Official Golang Documentation. (n.d.). *Effective Go - Slices and Arrays*. Retrieved from https://go.dev/doc/effective_go#arrays
4. GeeksforGeeks. (n.d.). *Sequential Search*. Retrieved from <https://www.geeksforgeeks.org/linear-search/>
5. GeeksforGeeks. (n.d.). *Binary Search*. Retrieved from <https://www.geeksforgeeks.org/binary-search/>
6. W3Schools. (n.d.). *Golang Structs Tutorial*. Retrieved from https://www.w3schools.com/go/go_struct.asp
7. Go.dev Blog. (n.d.). *Slices: Usage and Internals*. Retrieved from <https://go.dev/blog/slices-intro>