

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

MODUL 11

**PENCARIAN NILAI ACAK PADA
HIMPUNAN DATA**



Oleh:

ADINDA OLIVIA

2311102245

IF-11-07

S1 TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2024

I. DASAR TEORI

Pencarian Nilai Acak pada Himpunan Data

Pencarian nilai acak pada himpunan data adalah proses untuk menemukan atau memilih elemen secara acak dari sebuah kumpulan data yang ada. Proses ini dapat digunakan dalam berbagai konteks, seperti dalam algoritma pengujian, pemrograman berbasis probabilitas, atau dalam aplikasi yang membutuhkan pemilihan data secara tidak terduga.

Pada Golang, pencarian nilai acak pada himpunan data dapat dilakukan dengan menggunakan package `math/rand` untuk menghasilkan bilangan acak, dan kombinasi struktur data seperti `array`, `slice`, atau `map` untuk menyimpan data.

Konsep Dasar

1. **Pembuatan Data Himpunan:** Himpunan data pada Golang biasanya berupa `array` atau `slice`. `Array` adalah struktur data yang memiliki ukuran tetap, sementara `slice` memiliki ukuran yang bisa berubah. Kedua struktur data ini dapat digunakan untuk menyimpan nilai-nilai yang akan dipilih secara acak.
2. **Penggunaan Package `math/rand`:** Package `math/rand` menyediakan fungsi untuk menghasilkan angka acak. Fungsi yang paling umum digunakan untuk menghasilkan angka acak adalah `rand.Intn(n)` yang akan menghasilkan angka acak antara 0 dan $n-1$.
3. **Pemilihan Data Secara Acak:** Untuk memilih nilai acak dari himpunan data, pertama-tama kita perlu menghasilkan angka acak yang menjadi indeks dalam `array` atau `slice`. Angka acak ini akan digunakan untuk mengakses elemen data pada posisi tersebut.
4. **Fungsi `rand.Seed`** Sebelum memulai pencarian nilai acak, sangat penting untuk melakukan seeding pada generator angka acak menggunakan `rand.Seed()` agar nilai acak yang dihasilkan berbeda setiap kali program dijalankan. Jika tidak di-seed, generator akan menghasilkan angka acak yang sama setiap kali dijalankan.

Contoh Implementasi

Berikut adalah contoh sederhana pencarian nilai acak pada himpunan data menggunakan Golang:

```
package main

import (
```

```

    "fmt"

    "math/rand"

    "time"
)

func main() {

    // Menyimpan data dalam slice

    data := []int{10, 20, 30, 40, 50}


    // Menyiapkan angka acak

    rand.Seed(time.Now().UnixNano()) // Menggunakan waktu sebagai seed


    // Memilih indeks acak dari slice

    index := rand.Intn(len(data)) // Hasil acak antara 0 dan len(data)-
1

    // Menampilkan nilai acak

    fmt.Println("Nilai acak:", data[index])

}

```

Penjelasan Kode:

1. **Data:** Program ini menyimpan data dalam bentuk slice data yang berisi angka-angka.
2. **Seeding:** `rand.Seed(time.Now().UnixNano())` digunakan untuk menginisialisasi seed berdasarkan waktu saat ini, sehingga nilai acak yang dihasilkan berbeda pada setiap eksekusi.
3. **Pemilihan Acak:** `rand.Intn(len(data))` digunakan untuk menghasilkan indeks acak yang akan dipakai untuk mengakses nilai dalam slice.
4. **Output:** Program akan mencetak nilai acak yang dipilih dari slice.

Pencarian nilai acak sangat berguna dalam banyak aplikasi, termasuk:

- **Simulasi Monte Carlo:** Di mana hasil acak digunakan untuk memperkirakan hasil yang mungkin terjadi dalam suatu sistem kompleks.
- **Permainan atau Aplikasi Hiburan:** Menggunakan angka acak untuk menentukan hasil dari permainan atau hiburan lainnya.
- **Pemrograman Uji:** Di mana pemilihan data acak digunakan untuk menguji berbagai kemungkinan input dalam pengembangan perangkat lunak.

Kesimpulan

Pencarian nilai acak pada himpunan data menggunakan Golang melibatkan penggunaan package math/rand dan pemahaman dasar mengenai struktur data yang digunakan, seperti array atau slice. Proses ini memungkinkan pemrogram untuk memilih data secara acak dalam berbagai aplikasi, baik untuk keperluan pengujian, simulasi, atau hiburan.

II. GUIDED

a. Contoh 1

Source code

```
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
}

// 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
}

// 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)
    }

    // 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)
    }
}

```

Screenshoot program

```

57 func main() {
63     fmt.Scanln(&n)
64
65     // Input data mahasiswa
66     fmt.Println("Masukkan data mahasiswa:")
67     for i := 0; i < n; i++ {
68         fmt.Printf("Mahasiswa %d\n", i+1)
69         fmt.Print("Nama: ")
70         fmt.Scanln(&data[i].nama)
71         fmt.Print("NIM: ")
72         fmt.Scanln(&data[i].nim)
73         fmt.Print("Kelas: ")
74         fmt.Scanln(&data[i].kelas)
75         fmt.Print("Jurusan: ")
76         fmt.Scanln(&data[i].jurusan)
77         fmt.Print("IPK: ")
78         fmt.Scanln(&data[i].ipk)
79     }

```

PROBLEMS 10 OUTPUT DEBUG CONSOLE TERMINAL
 Mahasiswa 1
 Nama: adinda
 NIM: 2311102245
 Kelas: 11g
 Jurusan: if
 IPK: 3.7
 Mahasiswa 2
 Nama: natta
 NIM: 2311102240
 Kelas: 11d
 Jurusan: dkv
 IPK: 4.0
 Masukkan nama mahasiswa yang ingin dicari: natta
 Mahasiswa ditemukan pada indeks: 1
 Masukkan NIM mahasiswa yang ingin dicari: 2311102245
 Mahasiswa ditemukan pada indeks: 0
 PS C:\Users\victu>

Penjelasan: Program ini mendefinisikan struktur data mahasiswa dengan informasi nama, NIM, kelas, jurusan, dan IPK, lalu menyimpan data mahasiswa dalam array bertipe struct. Dua metode pencarian digunakan: **Sequential Search** untuk mencari mahasiswa berdasarkan nama, dan **Binary Search** untuk mencari mahasiswa berdasarkan NIM yang sudah terurut. Pada fungsi utama, pengguna diminta untuk memasukkan jumlah mahasiswa, kemudian data mahasiswa yang terdiri dari nama, NIM, kelas, jurusan, dan IPK. Setelah itu, program memungkinkan pencarian mahasiswa berdasarkan nama dan NIM, dengan hasil yang menunjukkan indeks mahasiswa jika ditemukan, atau pesan jika tidak ditemukan.

b. Contoh 2

Source code

```
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
}

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)
}
}

```

Screenshoot program

```

C:\Users\vicu> Documents > ALPRO 2 S3 > modul 11 > -02.go > ...
9 func BinarySearch_2(arrInt, n int, x int) int {
25     } else { // jika ditemukan
26         found = med
27     }
28 }
29 return found
30 }
31
32 func main() {
33     // Contoh penggunaan
34     var data arrInt
35     var n int
36     // Input jumlah elemen
37     fmt.Print("Masukkan jumlah elemen dalam array: ")
38     fmt.Scanln(&n)
39     // Input elemen array (harus terurut secara descending)
40     fmt.Println("Masukkan elemen array (harus terurut menurun):")
41     for i := 0; i < n; i++ {
42         fmt.Printf("Elemen %d: ", i+1)
43         fmt.Scanln(&data[i])
44     }
45
46     // Input elemen yang dicari
47     var search int
48     fmt.Print("Masukkan elemen yang ingin dicari: ")
49     fmt.Scanln(&search)

```

PROBLEMS 10 OUTPUT DEBUG CONSOLE TERMINAL

```

Masukkan jumlah elemen dalam array: 4
Masukkan elemen array (harus terurut menurun):
Elemen 1: 2
Elemen 2: 3
Elemen 3: 4
Elemen 4: 5
Masukkan elemen yang ingin dicari: 3
Elemen ditemukan pada indeks: 1
PS C:\Users\vicu>

```

Ln 61, Col 1 Tab Size: 4 UTF-8 CRLF Go 1.23.2

Penjelasan: Program ini mendefinisikan sebuah array `arrInt` yang dapat menampung 4321 elemen bertipe integer. Fungsi **Binary Search** yang digunakan mencari elemen dalam array yang sudah terurut secara menurun (descending). Fungsi ini akan mengembalikan indeks elemen yang dicari, atau -1 jika elemen tersebut tidak ditemukan. Dalam fungsi utama (main), pengguna diminta untuk memasukkan jumlah elemen dalam array dan kemudian mengisi array dengan

elemen-elemen yang harus terurut secara menurun. Setelah itu, pengguna dapat memasukkan elemen yang ingin dicari, dan hasil pencarian akan menunjukkan apakah elemen tersebut ada dalam array beserta indeksinya, atau memberi pesan bahwa elemen tidak ditemukan.

c. Contoh 3
Search code

```
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
}

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)
    }
}

```

Screenshot program

```

C:\Users\victu> Documents\ALPRO 2 S3> modul 11> go run .
24 func main() {
29     // Input jumlah elemen
30     fmt.Print("Masukkan jumlah elemen dalam array: ")
31     fmt.Scanln(&n)
32
33     // Input elemen array
34     fmt.Println("Masukkan elemen array:")
35     for i := 0; i < n; i++ {
36         fmt.Printf("Elemen %d: ", i+1)
37         fmt.Scanln(&data[i])
38     }
39     // Input elemen yang dicari
40     var search string
41     fmt.Print("Masukkan elemen yang ingin dicari: ")
42     fmt.Scanln(&search)
43     // Panggil fungsi sequential search
44     result := SeqSearch_1(data, n, search)
45     // Cetak hasil
46     if result == -1 {
47         fmt.Println("Elemen tidak ditemukan dalam array.")
48     } else {
49         fmt.Printf("Elemen ditemukan pada indeks: %d\n", result)
50     }
51 }
52
Masukkan elemen array:
Elemen 1: 1
Elemen 2: 2
Elemen 3: 3
Elemen 4: 4
Elemen 5: 5
Masukkan elemen yang ingin dicari: 5
Elemen ditemukan pada indeks: 4
PS C:\Users\victu>

```

Penjelasan: Program ini mendefinisikan sebuah array `arrStr` yang dapat menampung hingga 1234 elemen bertipe string. Fungsi **Sequential Search** digunakan untuk mencari elemen dalam array yang berisi `n` teks. Fungsi ini akan mengembalikan indeks elemen yang dicari jika ditemukan, atau `-1` jika elemen tersebut tidak ada dalam array.

Dalam fungsi utama (`main`), pengguna diminta untuk memasukkan jumlah elemen dalam array dan kemudian mengisi array dengan teks. Setelah itu, pengguna dapat memasukkan elemen yang ingin dicari dalam array. Program kemudian akan

memanggil fungsi **Sequential Search** dan mencetak hasilnya, yaitu indeks elemen yang ditemukan, atau memberi pesan bahwa elemen tidak ditemukan dalam array.

III. UNGUIDED

a. Soal nomor 1

Source code

```
package main

import (
    "fmt"
)

func main() {
    // Array untuk menyimpan jumlah suara untuk setiap calon (indeks
    0-19 untuk calon 1-20)
    var suara [20]int
    var totalSuara, suaraSah int
    var input int

    // Membaca input
    fmt.Println("Masukkan daftar suara (dalam format angka yang
    dipisahkan spasi), diakhiri dengan angka 0:")
    for {
        // Membaca angka input
        _, err := fmt.Scan(&input)
        if err != nil {
            fmt.Println("Input tidak valid!")
            break
        }

        // Jika input adalah 0, berhenti
        if input == 0 {
            break
        }

        // Memvalidasi suara yang hanya dalam rentang 1 hingga 20
        if input >= 1 && input <= 20 {
```

```

        suara[input-1]++ // Increment suara untuk calon yang
sesuai

        suaraSah++      // Increment jumlah suara sah
    }

    totalSuara++ // Increment total suara yang dimasukkan
}

// Menampilkan jumlah total suara dan suara sah
fmt.Printf("Suara masuk: %d\n", totalSuara)
fmt.Printf("Suara sah: %d\n", suaraSah)

// Menampilkan suara yang diterima oleh setiap calon yang valid
for i := 0; i < 20; i++ {
    if suara[i] > 0 {
        fmt.Printf("%d: %d\n", i+1, suara[i])
    }
}
}

```

Screenshot program

```

25     break
26 }
27
28 // Memvalidasi suara yang hanya dalam rentang 1 hingga 20
29 if input >= 1 && input <= 20 {
30     suara[input-1]++ // Increment suara untuk calon yang sesuai
31     suaraSah++      // Increment jumlah suara sah
32 }
33 totalSuara++ // Increment total suara yang dimasukkan
34 }
35
36 // Menampilkan jumlah total suara dan suara sah
37 fmt.Printf("Suara masuk: %d\n", totalSuara)
38 fmt.Printf("Suara sah: %d\n", suaraSah)
39
40 // Menampilkan suara yang diterima oleh setiap calon yang valid
41 for i := 0; i < 20; i++ {
42     if suara[i] > 0 {
43         fmt.Printf("%d: %d\n", i+1, suara[i])
44     }
45 }
46 }
47

```

PROBLEMS 10 OUTPUT DEBUG CONSOLE TERMINAL

```

2 1 2 3 3 4 1 2 5 -5 -1 0 1 2
Suara masuk: 11
Suara sah: 9
1: 2
2: 3
3: 2
4: 1
5: 1
PS C:\Users\victu>

```

Ln 40, Col 68 Tab Size: 4 UTF-8 CRLF Go 1.23.2

21:32 05/12/2024

Penjelasan: Program ini digunakan untuk menghitung jumlah suara dalam sebuah pemilihan dengan 20 calon. Pengguna diminta memasukkan suara sebagai angka, diakhiri dengan angka 0. Setiap angka yang dimasukkan antara 1 hingga 20 akan dihitung sebagai suara sah untuk calon yang sesuai. Program kemudian menghitung total suara yang dimasukkan dan jumlah suara sah. Setelah itu, program akan menampilkan jumlah suara untuk setiap calon yang mendapatkan suara, serta total suara yang sah dan total suara yang dimasukkan.

b. Soal nomor 2

Source code

```
package main

import (
    "fmt"
    "sort"
)

func main() {
    // Array untuk menyimpan jumlah suara untuk setiap calon (indeks
    // 0-19 untuk calon 1-20)
    var suara [20]int
    var totalSuara, suaraSah int
    var input int

    // Membaca input
    fmt.Println("Masukkan daftar suara (dalam format angka yang
    dipisahkan spasi), diakhiri dengan angka 0:")
    for {
        // Membaca angka input
        _, err := fmt.Scan(&input)
        if err != nil {
            fmt.Println("Input tidak valid!")
            break
        }

        // Jika input adalah 0, berhenti
        if input == 0 {
            break
        }

        // Memvalidasi suara yang hanya dalam rentang 1 hingga 20
        if input >= 1 && input <= 20 {
            suara[input-1]++ // Increment suara untuk calon yang
            sesuai
            suaraSah++ // Increment jumlah suara sah
        }
    }

    // Menampilkan hasil
    sort.Ints(suara)
    for i, v := range suara {
        if v > 0 {
            fmt.Printf("Calon %d: %d suara\n", i+1, v)
        }
    }
    fmt.Printf("Total suara sah: %d\n", suaraSah)
    fmt.Printf("Total suara yang dimasukkan: %d\n", totalSuara)
}
```

```

    }
    totalSuara++ // Increment total suara yang dimasukkan
}

// Menampilkan jumlah total suara dan suara sah
fmt.Printf("Suara masuk: %d\n", totalSuara)
fmt.Printf("Suara sah: %d\n", suaraSah)

// Membuat slice untuk menyimpan pasangan (calon, suara)
var suaraCalon []struct {
    nomor, suara int
}

for i := 0; i < 20; i++ {
    if suara[i] > 0 {
        suaraCalon = append(suaraCalon, struct {
            nomor, suara int
        }{nomor: i + 1, suara: suara[i]})
    }
}

// Mengurutkan berdasarkan suara terbanyak, jika sama berdasarkan
nomor calon yang lebih kecil
sort.Slice(suaraCalon, func(i, j int) bool {
    if suaraCalon[i].suara == suaraCalon[j].suara {
        return suaraCalon[i].nomor < suaraCalon[j].nomor
    }
    return suaraCalon[i].suara > suaraCalon[j].suara
})

// Menampilkan hasil ketua dan wakil ketua
if len(suaraCalon) > 0 {
    fmt.Printf("Ketua RT: %d\n", suaraCalon[0].nomor)
    if len(suaraCalon) > 1 {
        fmt.Printf("Wakil ketua: %d\n", suaraCalon[1].nomor)
    }
}
}

```


Screenshoot program

```
func main() {  
    suara[input-1]++ // Increment suara untuk calon yang sesuai  
    suaraSah++ // Increment jumlah suara sah  
    totalSuara++ // Increment total suara yang dimasukkan  
}  
  
// Menampilkan jumlah total suara dan suara sah  
fmt.Printf("Suara masuk: %d\n", totalSuara)  
fmt.Printf("Suara sah: %d\n", suaraSah)  
  
// Membuat slice untuk menyimpan pasangan (calon, suara)  
var suaraCalon []struct {  
    nomor, suara int  
}  
  
for i := 0; i < 20; i++ {  
    if suara[i] > 0 {  
        suaraCalon = append(suaraCalon, struct {  
            nomor, suara int  
        })(nomor: i + 1, suara: suara[i])  
    }  
}  
  
// Mengurutkan berdasarkan suara terbanyak, jika sama berdasarkan nomor calon yang lebih kecil  
sort.Slice(suaraCalon, func(i, j int) bool {  
    if suaraCalon[i].suara == suaraCalon[j].suara {  
        return suaraCalon[i].nomor < suaraCalon[j].nomor  
    }  
    return suaraCalon[i].suara > suaraCalon[j].suara  
})  
  
// Menampilkan hasil pemilihan  
for i := 0; i < len(suaraCalon); i++ {  
    fmt.Printf("%d\t%d\t%d\n", suaraCalon[i].nomor, suaraCalon[i].suara, totalSuara)  
}
```

PROBLEMS 10 OUTPUT DEBUG CONSOLE TERMINAL

Masukkan daftar suara (dalam format angka yang dipisahkan spasi), diakhiri dengan angka 0:
2 3 1 4 2 4 4 4 5 1 2 1 -1 -3 0 1 1
Suara masuk: 14
Suara sah: 12
Ketua RT: 4
Wakil ketua: 1
PS C:\Users\victu>

Penjelasan: Program ini digunakan untuk menghitung dan menampilkan hasil pemilihan ketua dan wakil ketua RT dari 20 calon berdasarkan suara yang diterima. Pengguna memasukkan daftar suara untuk setiap calon, diakhiri dengan angka 0. Suara yang dimasukkan hanya dihitung jika berada dalam rentang 1 hingga 20. Setelah semua suara dimasukkan, program menghitung total suara dan suara sah, kemudian mengurutkan calon berdasarkan jumlah suara yang diterima. Jika dua calon memiliki jumlah suara yang sama, calon dengan nomor lebih kecil akan diprioritaskan. Program kemudian menampilkan ketua RT dengan suara terbanyak dan wakil ketua RT dengan suara terbanyak kedua.

c. Soal nomor 3

Source code

```
package main  
  
import "fmt"  
  
const NMAX = 1000000  
  
var data [NMAX]int  
  
func main() {  
    var n, k int
```

```

// Membaca jumlah data n dan angka k yang dicari
fmt.Print("Masukkan jumlah data (n): ")
fmt.Scan(&n)
fmt.Print("Masukkan angka yang ingin dicari (k): ")
fmt.Scan(&k)

// Memanggil fungsi untuk mengisi array
fillArray(n)

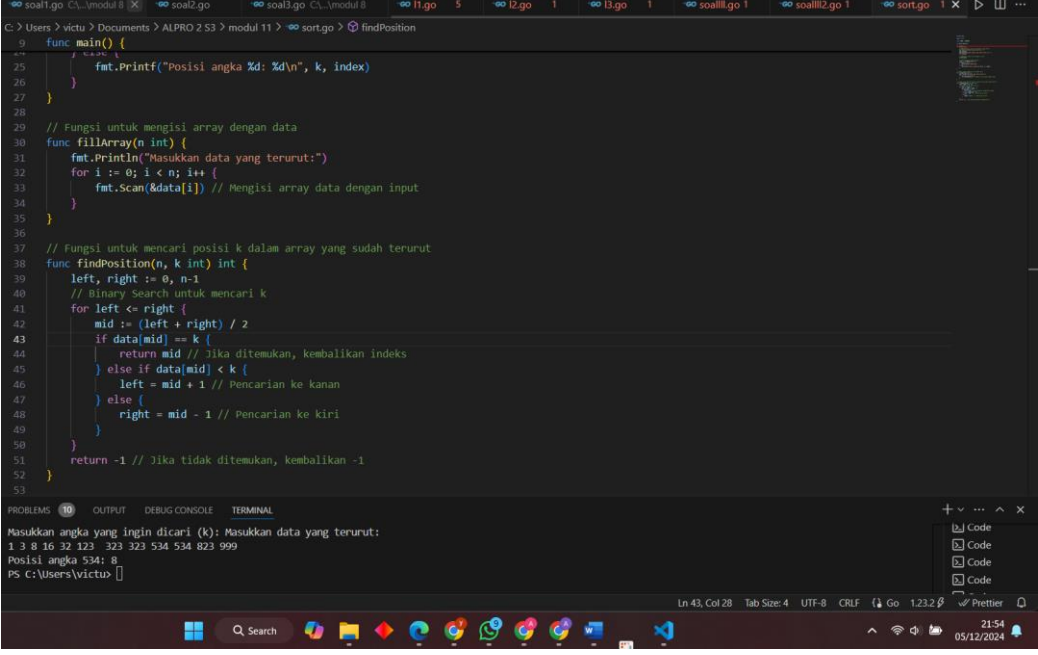
// Mencari posisi k dalam array
index := findPosition(n, k)
if index == -1 {
    fmt.Println("TIDAK ADA")
} else {
    fmt.Printf("Posisi angka %d: %d\n", k, index)
}
}

// Fungsi untuk mengisi array dengan data
func fillArray(n int) {
    fmt.Println("Masukkan data yang terurut:")
    for i := 0; i < n; i++ {
        fmt.Scan(&data[i]) // Mengisi array data dengan input
    }
}

// Fungsi untuk mencari posisi k dalam array yang sudah terurut
func findPosition(n, k int) int {
    left, right := 0, n-1
    // Binary Search untuk mencari k
    for left <= right {
        mid := (left + right) / 2
        if data[mid] == k {
            return mid // Jika ditemukan, kembalikan indeks
        } else if data[mid] < k {
            left = mid + 1 // Pencarian ke kanan
        } else {
            right = mid - 1 // Pencarian ke kiri
        }
    }
    return -1 // Jika tidak ditemukan, kembalikan -1
}

```

Screenshoot program



```
9 func main() {
25     fmt.Printf("Posisi angka %d: %d\n", k, index)
26 }
27 }
28
29 // Fungsi untuk mengisi array dengan data
30 func fillArray(n int) {
31     fmt.Println("Masukkan data yang terurut:")
32     for i := 0; i < n; i++ {
33         fmt.Scan(&data[i]) // Mengisi array data dengan input
34     }
35 }
36
37 // Fungsi untuk mencari posisi k dalam array yang sudah terurut
38 func findPosition(n, k int) int {
39     left, right := 0, n-1
40     // Binary Search untuk mencari k
41     for left <= right {
42         mid := (left + right) / 2
43         if data[mid] == k {
44             return mid // Jika ditemukan, kembalikan indeks
45         } else if data[mid] < k {
46             left = mid + 1 // Pencarian ke kanan
47         } else {
48             right = mid - 1 // Pencarian ke kiri
49         }
50     }
51     return -1 // Jika tidak ditemukan, kembalikan -1
52 }
53 }
```

PROBLEMS 10 OUTPUT DEBUG CONSOLE TERMINAL

Masukkan angka yang ingin dicari (k): Masukkan data yang terurut:
1 3 8 16 32 123 323 534 534 823 999
Posisi angka 534: 8
PS C:\Users\victu> |

Ln 43, Col 28 Tab Size: 4 UTF-8 CRLF 1.232 Prettier

Penjelasan: Program ini digunakan untuk mencari posisi sebuah angka dalam daftar data yang sudah terurut. Pengguna diminta untuk memasukkan jumlah data dan angka yang ingin dicari. Data yang dimasukkan akan disusun dalam array dan kemudian program menggunakan metode **Binary Search** untuk menemukan posisi angka tersebut dalam array. Jika angka ditemukan, program akan menampilkan posisi indeksinya, tetapi jika tidak ditemukan, program akan menampilkan pesan "TIDAK ADA". Binary Search bekerja dengan membagi array menjadi dua bagian secara berulang hingga angka yang dicari ditemukan atau dipastikan tidak ada dalam array.