LAPORAN PRAKTIKUM
ALGORITMA DAN PEMOGRAMAN 2

MODUL XII

PENGURUTAN DATA

Oleh:

YASVIN SYAHGANA

2311102065

S1 IF 11 02

S1 TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

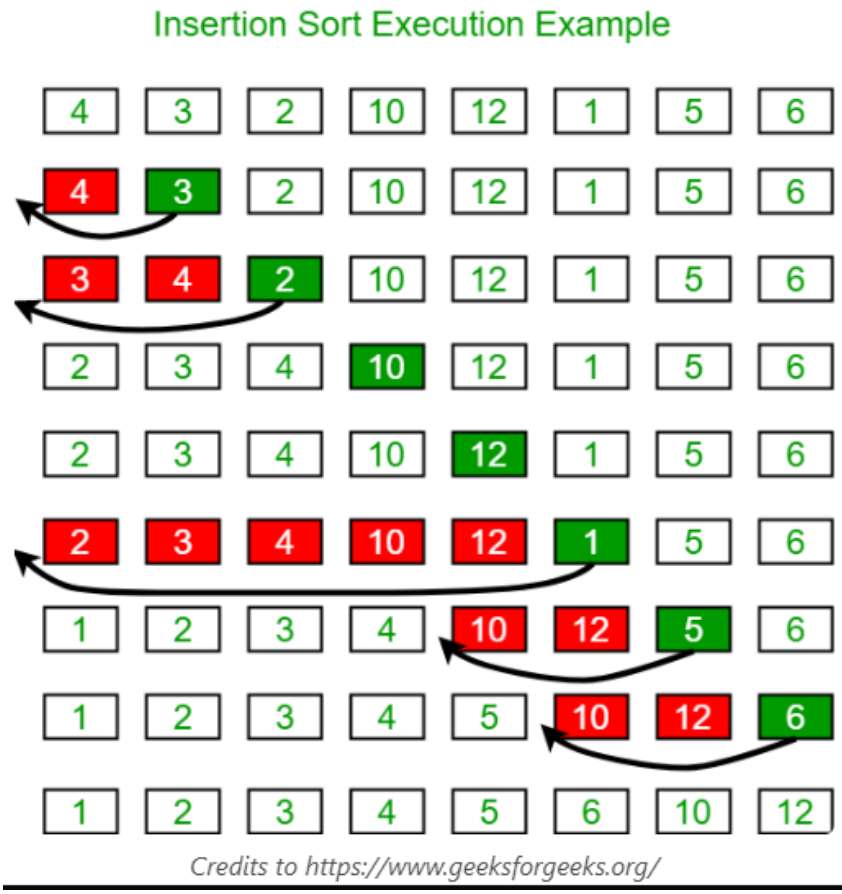
2024

I. DASAR TEORI

Pengurutan adalah proses mengatur data dalam format tertentu. Algoritme pengurutan menentukan cara untuk mengatur data dalam urutan tertentu yang dapat berupa urutan numerik atau leksikografis. Pentingnya pengurutan terletak pada kenyataan bahwa pencarian data dapat dioptimalkan ke tingkat yang sangat tinggi.

- Insertion Sort

  Insertion sort adalah algoritma pengurutan terkenal yang bekerja seperti mengurutkan setumpuk kartu di tangan. Saat Anda melanjutkan melalui elemen-elemen larik, Anda akan memindahkannya kembali sampai berada di tempat yang benar



Contoh Program

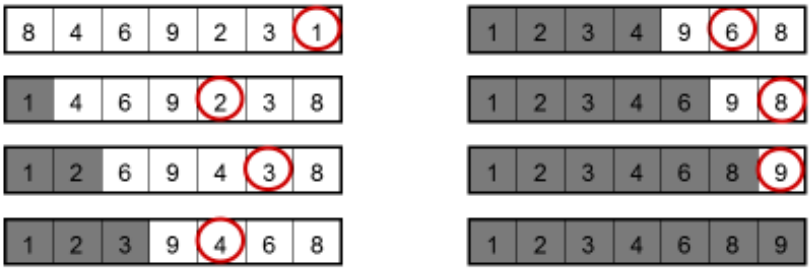
```
package main
```

```
import "fmt"

func main() {
  var n = []int{1, 39, 2, 9, 7, 54, 11}

  var i = 1
  for i < len(n) {
    var j = i
    for j >= 1 && n[j] < n[j - 1] {
      n[j], n[j - 1] = n[j - 1], n[j]

      j--
    }

    i++
  }

  fmt.Println(n)
}
```

- Selection Sort

  Pengurutan seleksi cukup menarik dan sederhana. Yang perlu Anda lakukan hanyalah mengganti elemen saat ini dalam iterasi dengan nilai terendah di sebelah kanan. Ketika Anda melangkah lebih jauh, bagian kiri larik akan diurutkan, dan Anda tidak perlu memeriksa elemen terakhir.

Selection Sort Example




```
package main
import "fmt"
func main() {
    var n = []int{1, 39, 2, 9, 7, 54, 11}
    var i = 1
    for i < len(n) - 1 {
        var j = i + 1
        var minIndex = i

        if j < len(n) {
            if n[j] < n[minIndex] {
                minIndex = j
            }
            j++
        }
        if minIndex != i {
            var temp = n[i]
            n[i] = n[minIndex]
            n[minIndex] = temp
        }
        i++
    }
    fmt.Println(n)
}
```

II. GUIDED

Guided 1 | Source Code + Screenshot hasil program beserta penjelasan

```
package main

import "fmt"

// Fungsi untuk mengurutkan array menggunakan selection
sort
func selectionSort(arr []int) {
     n := len(arr)
     for i := 0; i < n-1; i++ {
           maxIdx := i
           for j := i + 1; j < n; j++ {
                 if arr[j] < arr[maxIdx] { // Cari elemen
terbesar
                       maxIdx = j
                 }
           }
           arr[i], arr[maxIdx] = arr[maxIdx], arr[i] //
Tukar elemen
     }
}
func main() {
     var n int
     fmt.Print("Masukkan jumlah daerah (n): ")
     fmt.Scan(&n)
     if n <= 0 || n >= 1000 {
           fmt.Println("n harus lebih besar dari 0 dan
kurang dari 1000.")
           return
     }
     for i := 0; i < n; i++ {
           var m int
           fmt.Printf("Masukkan jumlah rumah kerabat
untuk daerah ke-%d: ", i+1)
           fmt.Scan(&m)
           if m <= 0 || m >= 1000000 {
```

```
                fmt.Println("m harus lebih besar dari 0
dan kurang dari 1000000.")
                return
            }
            // Masukkan nomor rumah
            houses := make([]int, m)
            fmt.Printf("Masukkan nomor rumah kerabat untuk
daerah ke-%d: ", i+1)
            for j := 0; j < m; j++ {
                fmt.Scan(&houses[j])
            }
            // Urutkan dengan selection sort
            selectionSort(houses)
            // Cetak hasil
            fmt.Printf("Hasil urutan rumah untuk daerah
ke-%d: ", i+1)
            for _, house := range houses {
                fmt.Printf("%d ", house)
            }
            fmt.Println()
        }
}
```

Output

```
PS D:\Pratikum Alpro 2\YASVIN SYAHGANA\Modul 12\Guided> go run "d:\Pratikum Alpro 2\YASVIN SYAHGANA\Modul 12\Guided\Guided1.go"
Masukkan jumlah daerah (n): 4
Masukkan jumlah rumah kerabat untuk daerah ke-1: 4
Masukkan nomor rumah kerabat untuk daerah ke-1: 2
42 32 12
Hasil urutan rumah untuk daerah ke-1: 2 12 32 42
Masukkan jumlah rumah kerabat untuk daerah ke-2: 3
Masukkan nomor rumah kerabat untuk daerah ke-2: 32 12 45
Hasil urutan rumah untuk daerah ke-2: 12 32 45
Masukkan jumlah rumah kerabat untuk daerah ke-3: 4
Masukkan nomor rumah kerabat untuk daerah ke-3: 37
7 6 8
Hasil urutan rumah untuk daerah ke-3: 6 7 8 37
Masukkan jumlah rumah kerabat untuk daerah ke-4: 6
Masukkan nomor rumah kerabat untuk daerah ke-4: 424 45 79 54 1 3
Hasil urutan rumah untuk daerah ke-4: 1 3 45 54 79 424
PS D:\Pratikum Alpro 2\YASVIN SYAHGANA\Modul 12\Guided>
```

Deskripsi

Program ini digunakan untuk mengurutkan data rumah per daerah dan disetiap daerah memiliki nomor rumah sesuai banyaknya rumah kerabat pada daerah tersebut. Cara kerja program ini adalah program akan meminta pengguna untuk menginputkan jumlah daerah (**n**) dengan sebuah kondisi `if n <= 0 || n >= 1000` jika memenuhi kondisi tersbut program akan berhenti atau tidak valid, setelah pengguna menginputkan (**n**) dengan tidak memenuhi kondisi diatas program kembali akan meminta pengguna untuk menginputkan jumlah rumah kerabat(**m**) yang juga memiliki kondisi `if m <= 0 || m >= 1000000` jika pengguna tidak memenuhi kondisi ini, program akan meminta pengguna untuk menginputkan nomor rumah dengan menggunakan array yang panjangnya sesuai dengan (**m**). Setelah pengguna menginputkan data, program akan menjalankan selection sort, dimana elemen terkecil dicari di bagian yang belum diurutkan dan ditukar dengan elemen pertama dari bagian tersebut, berulang hingga seluruh array terurut. Setelah proses pengurutan selesai maka program akan mencetak nomor rumah dari yang terkecil.

Guided 2 | Source Code + Screenshot hasil program beserta penjelasan

```
package main

import (
	"fmt"
	"math"
)

// Fungsi insertion sort untuk mengurutkan array
func insertionSort(arr []int) {
	n := len(arr)
	for i := 1; i < n; i++ {
		key := arr[i]
```

```
            j := i - 1
            // Geser elemen yang lebih besar dari key ke
kanan
            for j >= 0 && arr[j] > key {
                  arr[j+1] = arr[j]
                  j--
            }
            arr[j+1] = key
      }
}

// Fungsi untuk memeriksa apakah data berjarak tetap
func isDataConsistentlySpaced(arr []int) (bool, int) {
      if len(arr) < 2 {
            return true, 0 // Array dengan kurang dari 2
elemen dianggap berjarak tetap
      }
      // Hitung selisih awal
      diff := int(math.Abs(float64(arr[1] - arr[0])))
      for i := 1; i < len(arr)-1; i++ {
            currentDiff := int(math.Abs(float64(arr[i+1] -
                  arr[i])))
            if currentDiff != diff {
                  return false, 0 // Jika ada selisih yang
berbeda, tidak berjarak tetap
            }
      }
      return true, diff
}
func main() {
      var data []int
      var input int
      fmt.Println("Masukkan data (akhiri dengan bilangan
negatif):")
      for {
            fmt.Scan(&input)
            if input < 0 {
```

```
                break
            }
            data = append(data, input)
        }
        // Urutkan data menggunakan insertion sort
        insertionSort(data)
        // Periksa apakah data berjarak tetap
        isConsistent, diff := isDataConsistentlySpaced(data)
        // Cetak hasil
        fmt.Println("Hasil pengurutan:", data)
        if isConsistent {
            fmt.Printf("Data berjarak %d\n", diff)
        } else {
            fmt.Println("Data berjarak tidak tetap")
        }
}
```

Output

```
PS D:\Pratikum Alpro 2\YASVIN SYAHGANA\Modul 12\Guided> go run "d:\Pratikum Alpro 2\YASVIN SYAHGANA\Modul 12\Guided\Guided2.go"
Masukkan data (akhiri dengan bilangan negatif):
31 13 25 43 1 7 19 37 -5
Hasil pengurutan: [1 7 13 19 25 31 37 43]
Data berjarak 6
PS D:\Pratikum Alpro 2\YASVIN SYAHGANA\Modul 12\Guided> go run "d:\Pratikum Alpro 2\YASVIN SYAHGANA\Modul 12\Guided\Guided2.go"
Masukkan data (akhiri dengan bilangan negatif):
4 40 14 8 26 1 38 2 32 -31
Hasil pengurutan: [1 2 4 8 14 26 32 38 40]
Data berjarak tidak tetap
PS D:\Pratikum Alpro 2\YASVIN SYAHGANA\Modul 12\Guided>
```

Deskripsi

Program ini dibuat untuk mengurutkan sebuah data dengan menggunakan insertion sort dari inputan pengguna secara berulang ulang hingga memenuhi kondisi berhenti yaitu ketika inputan pengguna kurang dari 0 atau negatif dan program ini juga menghitung jarak antar bilangan bulat apakah sama jika sama program akan menampilkan output data tersebut berjarak **n**. Cara kerja program ini adalah pengguna akan menginputkan data yang berupa bilangan bulat dan dimasukkan kedalam sebuah array, pengguna akan terus menginputkan data hingga memenuhi kondisi untuk berhenti yaitu inputan pengguna kurang dari 0, Kemudian program akan

menjalakan insertion sort, insertion sort bekerja dengan mengambil elemen dari bagian yang belum terurut dan menyisipkannya ke posisi yang sesuai dalam bagian array yang sudah terurut, dengan cara menggeser elemen-elemen yang lebih besar ke kanan. Setelah berhasil diurutkan program akan memeriksa setiap selisih antara elemen-elemen berurutan dalam array yang telah diurutkan, jika semuanya berurutan sama atau true, maka program akan menampilkan return hasil dari selisih antar bilangan, akan tetapi jika salah satu atau banyaknya data tidak memiliki selisih sama atau false, maka program akan menampilkab bahwa data tersebut tidak berjarak sama.

III. UNGUIDED

Unguided 1 || Source Code + Screenshot hasil program beserta penjelasan

```
package main

import "fmt"

func GanjilSort(nomor []int) {
	for i := 0; i < len(nomor)-1; i++ {
		Max := i
		for j := i + 1; j < len(nomor); j++ {
			if nomor[j] < nomor[Max] {
				Max = j
			}
		}
		nomor[i], nomor[Max] = nomor[Max], nomor[i]
	}
}

func GenapSort(nomor []int) {
	for i := 0; i < len(nomor)-1; i++ {
		min := i
		for j := i + 1; j < len(nomor); j++ {
			if nomor[j] > nomor[min] {
				min = j
			}
		}
		nomor[i], nomor[min] = nomor[min], nomor[i]
	}
}

func main() {
	var n, m int

	fmt.Print("Inputkan banyaknya daerah: ")
	fmt.Scan(&n)
	if n <= 0 || n >= 1000 {
		fmt.Println("n harus lebih besar dari 0 dan
kurang dari 1000.")
		return
	}

	for i := 0; i < n; i++ {
		var nmrGanjil, nmrGenap []int
		fmt.Print("Jumlah rumah kerabat: ")
		fmt.Scan(&m)

		if m <= 0 || m >= 1000000 {
			fmt.Println("m harus lebih besar dari 0
dan kurang dari 1000000.")
			return
```

```
            }

            nomor := make([]int, m)
            for j := 0; j < m; j++ {
                  fmt.Scan(&nomor[j])
            }

            for _, x := range nomor {
                  if x%2 == 0 {
                        nmrGenap = append(nmrGenap, x)
                  } else {
                        nmrGanjil = append(nmrGanjil, x)
                  }
            }

            GanjilSort(nmrGanjil)
            GenapSort(nmrGenap)

            fmt.Print("Urutan rumah: ")
            for _, rmhGanjil := range nmrGanjil {
                  fmt.Print(rmhGanjil, " ")
            }
            for _, rmhGenap := range nmrGenap {
                  fmt.Print(rmhGenap, " ")
            }
            fmt.Println()
      }
}
```

Screenshoot Output

```
PS D:\Pratikum Alpro 2\YASVIN SYAHGANA\Modul 12\Unguided1> go run "d:\Pratikum Alpro 2\YASVIN SYAHGANA\Modul 12\Unguided1\unguided1.go"
Inputkan banyaknya daerah: 3
Jumlah rumah kerabat: 5
2 1 7 9 13
Urutan rumah: 1 7 9 13 2
Jumlah rumah kerabat: 6
189 15 27 39 75 133
Urutan rumah: 15 27 39 75 133 189
Jumlah rumah kerabat: 3
4 9 1
Urutan rumah: 1 9 4
PS D:\Pratikum Alpro 2\YASVIN SYAHGANA\Modul 12\Unguided1>
```

Deskripsi Program

Program ini dibuat untuk mengurutkan sebuah data dari inputan pengguna akan tetapi outputnya akan dibagi menjadi dua yaitu ganjil dan genap, output ganjil akan dari terkecil ke terbesar(*Ascending*) dan yang genap dari yang terbesar ke terkecil (*Descending*). Program ini memiliki penyimpanan

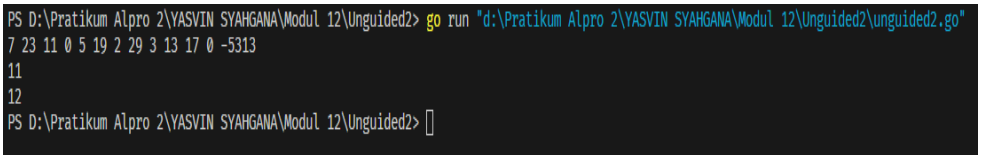
array sementara yang panjangnya sesuai m, dan m memiliki kondisi `if m <= 0 || m >= 1000000` jika tidak memenuhi kondisi tersebut program akan berlangsung, dan pengguna akan menginputkan nomor setiap rumah dan jumlah rumah tidak lebih dari m. Setelah diinputkan program akan memilah atau membedahkan menjadi dua array, yaitu array `var nmrGanjil, nmrGenap []int` kedua *value* dari array tersebut berupa nilai dari modulus array sementara diawal jika dibagi 2 akan sama dengan 0, yang berarti genap akan dimasukkan ke array genap, begitu juga ganjil akan dimasukkan ke ganjil. Kemudian program akan mensorting dengan sorting yang berbeda, ganjil akan disorting *Ascending* dan genap akan disorting *Descending*. Kemudian program akan mengeprint nilai array ganjil terlebih dahulu, kemudian array genap.

Unguided 2 || Source Code + Screenshot hasil program beserta penjelasan

```
package main

import "fmt"

func InsertionSorting(median []int) {
	for i := 0; i < len(median); i++ {
		cur := median[i]
		j := i - 1
		for j >= 0 && median[j] > cur {
			median[j+1] = median[j]
			j--
		}
		median[j+1] = cur
	}
}

func HitungMedian(median []int) float64 {
	InsertionSorting(median)
	n := len(median)
	if n%2 == 1 {
		return float64(median[n/2])
	} else {
		return float64(median[n/2-1]+median[n/2]) / 2
	}
}
func main() {
	var n int
```

```
	var median []int
	for {
		fmt.Scan(&n)

		if n == -5313 {
			break
		}

		if n == 0 {
			fmt.Println(HitungMedian(median))
		} else {
			median = append(median, n)
		}

	}
}
```

Screenshoot Output



Deskripsi Program

Program di atas dibuat untuk mencari sebuah nilai tengah dari inputan pengguna akan tetapi mencari nilai tengah tersebut berdasarkan sebuah parameter atau inputan pengguna yang berupa (0), jika pengguna menginputkan (0), maka program akan menjalankan `func HitungMedian(median []int) float64` didalam fungsi tersebut terdapat fungsi lain yaitu `func InsertionSorting(median []int)` yang digunakan untuk mengurutkan data, karena untuk mencari sebuah median memerlukan sebuah pengurutan data dari yang terkecil hingga terbesar. Cara kerja program ini adalah, Program memiliki inputan yang tidak terbatas apabila tidak memenuhi kondisi untuk keluar, pengguna akan menginputkan sebuah bilangan bulat dan program akan menjalankan `func HitungMedian(median []int) float64` dan fungsi tersebut juga menjalankan sorting yang dimana mengurutkan data inputan pengguna dari awal hingga 0(inputan). Setelah mensorting program akan mencari data median atau nilai tengah dari data yang ada. Program akan berjalan terus menerus hingga memenuhi kondisi keluar yaitu `n == -5313`.

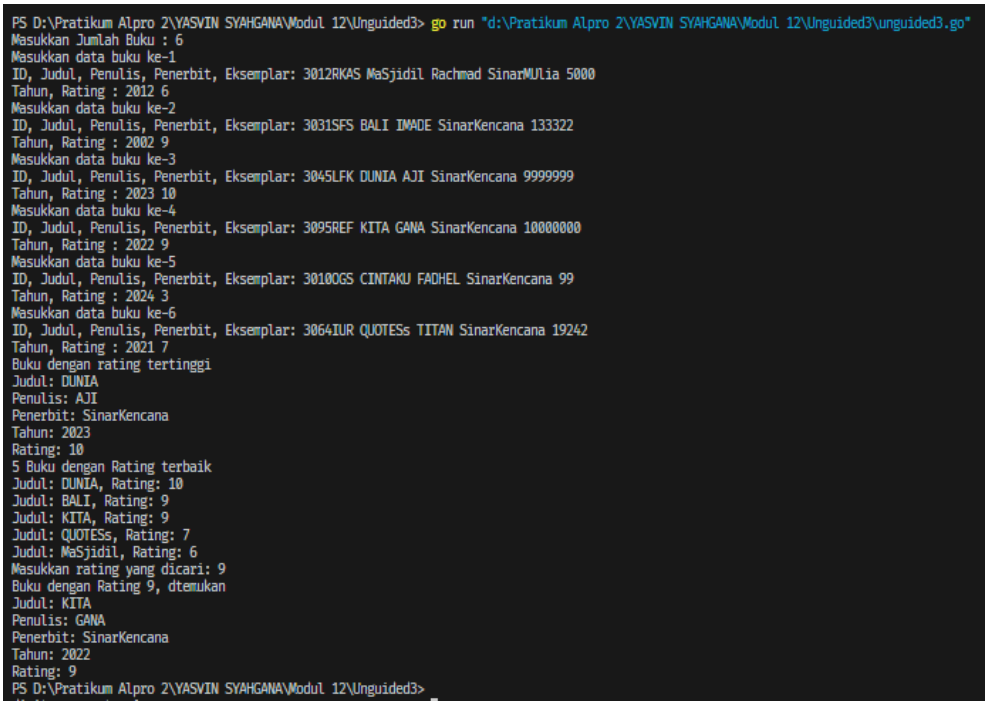
Unguided 3 || Source Code + Screenshot hasil program beserta penjelasan

```
package main

import "fmt"

const nMax = 7919

type Buku struct {
     id, judul, penulis, penerbit string
     eksemplar, tahun, rating     int
}

type DaftarBuku []Buku

func DaftarkanBuku(Pustaka *DaftarBuku, n int) {
     for i := 0; i < n; i++ {
           var buku Buku
           fmt.Printf("Masukkan data buku ke-%d\n", i+1)
           fmt.Print("ID, Judul, Penulis, Penerbit,
Eksemplar: ")
           fmt.Scan(&buku.id, &buku.judul, &buku.penulis,
&buku.penerbit, &buku.eksemplar)
           fmt.Print("Tahun, Rating : ")
           fmt.Scan(&buku.tahun, &buku.rating)
           *Pustaka = append(*Pustaka, buku)
     }
}

func CetakTerFavorit(Pustaka DaftarBuku) {
     terFavorit := Pustaka[0]

     for _, buku := range Pustaka {
           if buku.rating > terFavorit.rating {
                 terFavorit = buku
           }
     }
     fmt.Println("Buku dengan rating tertinggi ")
     fmt.Printf("Judul: %s\n", terFavorit.judul)
     fmt.Printf("Penulis: %s\n", terFavorit.penulis)
     fmt.Printf("Penerbit: %s\n", terFavorit.penerbit)
     fmt.Printf("Tahun: %d\n", terFavorit.tahun)
     fmt.Printf("Rating: %d\n", terFavorit.rating)
}

func UrutBuku(Pustaka *DaftarBuku) {
     n := len(*Pustaka)
     for i := 1; i < n; i++ {
           cur := (*Pustaka)[i]
           j := i - 1
```

```
            for j >= 0 && (*Pustaka)[j].rating <
cur.rating {
                  (*Pustaka)[j+1] = (*Pustaka)[j]
                  j--
            }
            (*Pustaka)[j+1] = cur
      }
}

func Cetak5Terbaru(Pustaka DaftarBuku) {
      if len(Pustaka) < 5 {
            fmt.Println("Data pustaka kurang dari 5
buku.")
            return
      }

      fmt.Println("5 Buku dengan Rating terbaik")
      for i := 0; i < 5; i++ {
            buku := Pustaka[i]
            fmt.Printf("Judul: %s, Rating: %d\n",
buku.judul, buku.rating)
      }
}

func CariBuku(Pustaka DaftarBuku, r int) {
      UrutBuku(&Pustaka)

      left, right := 0, len(Pustaka)-1
      for left <= right {
            mid := (left + right) / 2
            if Pustaka[mid].rating == r {
                  fmt.Printf("Buku dengan Rating %d,
dtemukan\n", r)
                  fmt.Printf("Judul: %s\n",
Pustaka[mid].judul)
                  fmt.Printf("Penulis: %s\n",
Pustaka[mid].penulis)
                  fmt.Printf("Penerbit: %s\n",
Pustaka[mid].penerbit)
                  fmt.Printf("Tahun: %d\n",
Pustaka[mid].tahun)
                  fmt.Printf("Rating: %d\n",
Pustaka[mid].rating)
                  return
            } else if Pustaka[mid].rating < r {
                  right = mid - 1
            } else {
                  left = mid + 1
            }
      }

}
```

```
 func main() {
      var Pustaka DaftarBuku
      var n, r int

      fmt.Print("Masukkan Jumlah Buku : ")
      fmt.Scan(&n)

      DaftarkanBuku(&Pustaka, n)
      CetakTerFavorit(Pustaka)
      UrutBuku(&Pustaka)
      Cetak5Terbaru(Pustaka)

      fmt.Print("Masukkan rating yang dicari: ")
      fmt.Scan(&r)
      CariBuku(Pustaka, r)
}
```

Screenshoot Output



Deskripsi Program

Program ini dibuat untuk mendata sebuah buku oleh pengguna, dan kemudian program akan mengembalikan sebuah buku dengan rating terbaik dan rating 5 terbaik. Kemudian program juga bisa menampilkan sebuah buku dengan mencari sebuah rating yang diingkan pengguna. Program ini memiliki struct yang berupa `type Buku struct` dan program ini juga

memiliki sebuah fungsi yang memiliki kegunaannya masing masing. Berikut beberapa fungsi yang ada :

- `func DaftarkanBuku(Pustaka *DaftarBuku, n int)`
  Fungsi ini digunakan untuk menambah buku ke dalam daftar pustaka, fungsi ini akan meminta user untuk menginputkan **n** banyaknya buku, dan kemudian program meminta pengguna untuk menginputkan id, judul, penulis, penerbit, eksemplar, tahun, rating
- `func CetakTerFavorit(Pustaka DaftarBuku)`
  Fungsi ini digunakan untuk menmapilkan buku dengan rating terbaik
- `func UrutBuku(Pustaka *DaftarBuku)`
  Fungsi ini digunakan untuk merapikan sebuah buku dengan cara mensorting semua buku yang ada dengan insertion sort
- `func Cetak5Terbaru(Pustaka DaftarBuku)`
  Fungsi ini digunakan untuk mencetak 5 rating terbaik dari daftar pustaka
- `func CariBuku(Pustaka DaftarBuku, r int)`
  Fungsi ini dibuat untuk mencari buku berdasarkan rating