

LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2
MODUL 12
PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA



Oleh:

M. AZKA HERMAWAN

2311102230

IF-11-02

S1 TEKNIK INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

A. Ide Algoritma Selection Sort

Pengurutan secara seleksi ini idenya adalah mencari nilai ekstrim pada sekumpulan data, kemudian meletakkan pada posisi yang seharusnya. Pada penjelasan berikut ini data akan diurut membesar (*ascending*), dan data dengan indeks kecil ada di "kiri" dan indeks besar ada di "kanan".

1. Cari nilai terkecil di dalam rentang data tersisa
2. Pindahkan/tukar tempat dengan data yang berada pada posisi paling kiri pada rentang data tersisa tersebut.
3. Ulangi proses ini sampai tersisa hanya satu data saja.

Algoritma ini dikenal juga dengan nama **Selection Sort**, yang mana pada algoritma ini melibatkan dua proses yaitu pencarian indeks nilai ekstrim dan proses pertukaran dua nilai atau *swap*.

	Notasi Algoritma	Notasi dalam bahasa Go
1	$i \leftarrow 1$	$i = 1$
2	while $i \leq n-1$ do	for $i \leq n-1$ {
3	$idx_min \leftarrow i - 1$	$idx_min = i - 1$
4	$j \leftarrow i$	$j = i$
5	while $j < n$ do	for $j < n$ {
6	if $a[idx_min] > a[j]$ then	if $a[idx_min] > a[j]$ {
7	$idx_min \leftarrow j$	$idx_min = j$
8	endif	}
9	$j \leftarrow j + 1$	$j = j + 1$
10	endwhile	}
11	$t \leftarrow a[idx_min]$	$t = a[idx_min]$
12	$a[idx_min] \leftarrow a[i-1]$	$a[idx_min] = a[i-1]$
13	$a[i-1] \leftarrow t$	$a[i-1] = t$
14	$i \leftarrow i + 1$	$i = i + 1$
15	endwhile	}

B. Algoritma Selection Sort

Adapun algoritma *selection sort* pada untuk mengurutkan array bertipe data bilangan bulat secara membesar atau ascending adalah sebagai berikut ini!

```

..   ...
5   type arrInt [4321]int
..   ...
15  func selectionSort1(T *arrInt, n int){
16  /* I.S. terdefinisi array T yang berisi n bilangan bulat
17     F.S. array T terurut secara ascending atau membesar dengan SELECTION SORT */
18     var t, i, j, idx_min int
19

```

```

20     i = 1
21     for i <= n-1 {
22         idx_min = i - 1
23         j = i
24         for j < n {
25             if T[idx_min] > T[j] {
26                 idx_min = j
27             }
28             j = j + 1
29         }
30         t = T[idx_min]
31         T[idx_min] = T[i-1]
32         T[i-1] = t
33         i = i + 1
34     }
35 }

```

Sama halnya apabila array yang akan diurutkan adalah bertipe data struct, maka tambahkan field pada saat proses perbandingan nilai ekstrim, kemudian tipe data dari variabel *t* sama dengan struct dari arraynya.

```

..   ...
5   type mahasiswa struct {
..     nama, nim, kelas, jurusan string
..     ipk float64
.. }
.. type arrMhs [2023]mahasiswa
..   ...
15  func selectionSort2(T * arrMhs, n int){
16  /* I.S. terdefinisi array T yang berisi n data mahasiswa
17     F.S. array T terurut secara ascending atau membesar berdasarkan ipk dengan
18     menggunakan algoritma SELECTION SORT */
19     var i, j, idx_min int
20     var t mahasiswa
21     i = 1
22     for i <= n-1 {
23         idx_min = i - 1
24         j = i
25         for j < n {
26             if T[idx_min].ipk > T[j].ipk {
27                 idx_min = j
28             }
29             j = j + 1
30         }
31         t = T[idx_min]
32         T[idx_min] = T[i-1]
33         T[i-1] = t
34         i = i + 1
35     }
36 }

```

C. Ide Algoritma Insertion Sort

Pengurutan secara insertion ini idenya adalah menyisipkan suatu nilai pada posisi yang seharusnya. Berbeda dengan pengurutan seleksi, yang mana pada pengurutan ini tidak dilakukan pencarian nilai ekstrim terlebih dahulu, cukup memilih suatu nilai tertentu kemudian mencari posisinya secara *sequential search*. Pada penjelasan berikut ini data akan diurut mengecil (*descending*), dan data dengan indeks kecil ada di "kiri" dan indeks besar ada di "kanan".

1. Untuk satu data yang belum terurut dan sejumlah data yang sudah diurutkan:
Geser data yang sudah terurut tersebut (ke kanan), sehingga ada satu ruang kosong untuk memasukkan data yang belum terurut ke dalam data yang sudah terurut dan tetap menjaga keterurutan.
2. Ulangi proses tersebut untuk setiap data yang belum terurut terhadap rangkaian data yang sudah terurut.

Algoritma ini dikenal juga dengan nama ***Insertion Sort***, yang mana pada algoritma ini melibatkan dua proses yaitu pencarian sekuensial dan penyisipan.

	Notasi Algoritma	Notasi dalam bahasa Go
1	$i \leftarrow 1$	$i = 1$
2	while $i \leq n-1$ do	for $i \leq n-1$ {
3	$j \leftarrow i$	$j = i$
4	$temp \leftarrow a[j]$	$temp = a[j]$
5	while $j > 0$ and $temp > a[j-1]$ do	for $j > 0 \ \&\& \ temp > a[j-1]$ {
6	$a[j] \leftarrow a[j-1]$	$a[j] = a[j-1]$
7	$j \leftarrow j - 1$	$j = j - 1$
8	endwhile	}
9	$a[j] \leftarrow temp$	$a[j] = temp$
10	$i \leftarrow i + 1$	$i = i + 1$
11	endwhile	}

D. Algoritma Insertion Sort

Adapun algoritma *insertion sort* pada untuk mengurutkan array bertipe data bilangan bulat secara mengecil atau *descending* adalah sebagai berikut ini!

```

..    ...
5   type arrInt [4321]int
..    ...
15  func insertionSort1(T *arrInt, n int){
16  /* I.S. terdefinisi array T yang berisi n bilangan bulat
17     F.S. array T terurut secara mengecil atau descending dengan INSERTION SORT*,
18     var temp, i, j int
19     i = 1
20     for i <= n-1 {
21         j = i
22         temp = T[j]
23         for j > 0 && temp > T[j-1] {
24             T[j] = T[j-1]
25             j = j - 1
26         }
27         T[j] = temp
28         i = i + 1
29     }
30 }

```

Sama halnya apabila array yang akan diurutkan adalah bertipe data struct, maka tambahkan field pada saat proses perbandingan dalam pencarian posisi, kemudian tipe data dari variabel **temp** sama dengan struct dari arraynya.

```

..    ...
5   type mahasiswa struct {
..     nama, nim, kelas, jurusan string
..     ipk float64
.. }
.. type arrMhs [2023]mahasiswa
..    ...
15  func insertionSort2(T * arrMhs, n int){
16  /* I.S. terdefinisi array T yang berisi n data mahasiswa
17     F.S. array T terurut secara mengecil atau descending berdasarkan nama dengan
18     menggunakan algoritma INSERTION SORT */
19     var temp i, j int
20     var temp mahasiswa
21     i = 1
22     for i <= n-1 {
23         j = i
24         temp = T[j]
25         for j > 0 && temp.nama > T[j-1].nama {
26             T[j] = T[j-1]
27             j = j - 1
28         }
29         T[j] = temp
30         i = i + 1

```

II. GUIDED

GUIDED I

Source Code

```
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()
        }
    }
}

```

Screenshot

```
PS C:\2311102230_M. AZKA HERMAWAN_MODUL-12> go run "c:\2311102230_M. AZKA H
Masukkan jumlah daerah (n): 3
Masukkan jumlah rumah kerabat untuk daerah ke-1: 3
Masukkan nomor rumah kerabat untuk daerah ke-1: 2 3 4
Hasil urutan rumah untuk daerah ke-1: 4 3 2
```

Deskripsi:

Program tersebut merupakan program yang berfungsi untuk mengurutkan nomor rumah kerabat di beberapa daerah menggunakan metode Selection Sort.

Program ini meminta pengguna untuk memasukkan data sejumlah daerah dan nomor rumah kerabat yang ada di setiap daerah. Setelah data nomor rumah dimasukkan, program akan mengurutkannya dari yang terbesar hingga terkecil menggunakan algoritma Selection Sort.

Fungsi selection sort pada program berfungsi untuk mengurutkan array.

GUIDED II

Source Code

```
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)
    }

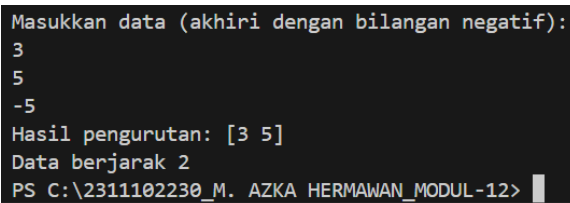
    // Urutkan data menggunakan insertion sort
    insertionSort(data)

    // 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")
    }
}

```

Screenshot



```

Masukkan data (akhiri dengan bilangan negatif):
3
5
-5
Hasil pengurutan: [3 5]
Data berjarak 2
PS C:\2311102230_M. AZKA HERMAWAN_MODUL-12>

```

Deskripsi:

Program ini berfungsi untuk menerima input angka dari pengguna, mengurutkan angka-angka tersebut menggunakan algoritma insertion sort, kemudian memeriksa apakah angka-angka yang telah diurutkan memiliki jarak yang konsisten atau tetap antara satu angka dengan angka berikutnya.

III. UNGUIDED

UNGUIDED I

Source Code

```
package main

import (
    "fmt"
    "sort"
)

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
        }

        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])
        }

        var odd, even []int
        for _, house := range houses {
            if house%2 == 0 {
                even = append(even, house)
            } else {
                odd = append(odd, house)
            }
        }

        sort.Ints(odd)
        sort.Ints(even)

        fmt.Printf("Hasil urutan rumah untuk daerah ke-%d:
", i+1)
        for _, house := range odd {
            fmt.Printf("%d ", house)
        }
        for _, house := range even {
            fmt.Printf("%d ", house)
        }
        fmt.Println()
    }
}

```

Screenshot

```
PS C:\2311102230_M. AZKA HERMAWAN_MODUL-12> go run "c:\2311102230_M. AZKA HERMAWAN_MODUL-12\unguided 1\1.go"
Masukkan jumlah daerah (n): 2
Masukkan jumlah rumah kerabat untuk daerah ke-1: 3
Masukkan nomor rumah kerabat untuk daerah ke-1: 24
23
32
Hasil urutan rumah untuk daerah ke-1: 23 24 32
Masukkan jumlah rumah kerabat untuk daerah ke-2: 5
Masukkan nomor rumah kerabat untuk daerah ke-2: 1
2
3
4
5
Hasil urutan rumah untuk daerah ke-2: 1 3 5 2 4
PS C:\2311102230_M. AZKA HERMAWAN_MODUL-12>
```

Deskripsi:

Program ini berguna untuk mengelompokkan dan mengurutkan data nomor rumah di beberapa daerah, yang bisa diterapkan dalam berbagai kasus seperti analisis wilayah berdasarkan data rumah atau pengaturan logistik di suatu daerah.

UNGUIDED 2

Source Code

```
package main

import "fmt"

func hitungMedian(data []int, panjang int) float64 {
    if panjang%2 == 1 {
        return float64(data[panjang/2])
    }
    return float64(data[panjang/2-1]+data[panjang/2]) /
2.0
}

func SelectionSort(data []int, panjang int) {
    for i := 0; i < panjang-1; i++ {
        minimal := i
        for j := i; j < panjang; j++ {
            if data[j] < data[minimal] {
                minimal = j
            }
        }
        data[i], data[minimal] = data[minimal], data[i]
    }
}

func main() {
    var input int
    var data [1000000]int
    var semuaMedian [1000000]float64
    panjang := 0
    totalMedian := 0

    fmt.Println("Masukan bilangan : ")

    for {
```

```
        fmt.Scan(&input)

        if input == -5313 {
            break
        }

        if input == 0 {
            if panjang > 0 {
                SelectionSort(data[:], panjang)

                median := hitungMedian(data[:], panjang)
                semuaMedian[totalMedian] = median
                totalMedian++
            }
        } else {
            data[panjang] = input
            panjang++
        }
    }

    fmt.Println("\nHasil median: ")
    for i := 0; i < totalMedian; i++ {
        fmt.Printf("Median %d: %.0f\n", i+1,
semuaMedian[i])
    }
    fmt.Println()
}
```


Screenshot

```
Masukan bilangan :  
7 23 11 0 5 19 2 29 3 13 17 0 -5313  
  
Hasil median:  
Median 1: 11  
Median 2: 12  
  
PS C:\2311102230_M. AZKA HERMAWAN_MODUL-12>
```

Deskripsi:

Program ini untuk menghitung median dari sejumlah bilangan yang dimasukkan oleh pengguna. Program ini juga menyimpan semua nilai median yang telah dihitung dan menampilkannya setelah input selesai.

UNGUIDED 3

Source Code

```
package main

import (
    "fmt"
)

const nMax = 7919

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

type DaftarBuku struct {
    pustaka []Buku
}

func DaftarkanBuku(pustaka *DaftarBuku, buku Buku) {
    if len(pustaka.pustaka) < nMax {
        pustaka.pustaka = append(pustaka.pustaka, buku)
    }
}

func CetakTerfavorit(pustaka DaftarBuku, n int) {
    if n == 0 {
```

```

        fmt.Println("Tidak ada buku dalam pustaka.")
        return
    }

    terfavorit := pustaka.pustaka[0]
    for i := 1; i < n; i++ {
        if pustaka.pustaka[i].rating > terfavorit.rating {
            terfavorit = pustaka.pustaka[i]
        }
    }

    fmt.Println("Buku Terfavorit:")
    fmt.Printf("Judul          : %s \nPenulis      : %s\nPenerbit    : %s\nTahun        : %d\nRating       : %d\n\n",
        terfavorit.judul, terfavorit.penulis,
        terfavorit.penerbit, terfavorit.tahun, terfavorit.rating)
}

func UrutBuku(pustaka *DaftarBuku, n int) {
    for i := 1; i < n; i++ {
        key := pustaka.pustaka[i]
        j := i - 1
        for j >= 0 && pustaka.pustaka[j].rating <
key.rating {
            pustaka.pustaka[j+1] = pustaka.pustaka[j]
            j--
        }
        pustaka.pustaka[j+1] = key
    }
}

func Cetak5Terbaru(pustaka DaftarBuku, n int) {
    if n == 0 {
        fmt.Println("Tidak ada buku dalam pustaka.")
        return
    }
}

```

```

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

func CariBuku(pustaka DaftarBuku, n, r int) {
    UrutBuku(&pustaka, n)

    low, high := 0, n-1
    for low <= high {
        mid := (low + high) / 2
        if pustaka.pustaka[mid].rating == r {
            buku := pustaka.pustaka[mid]
            fmt.Printf("Buku ditemukan:\nJudul          :
%s\nPenulis      : %s\nPenerbit    : %s\nTahun        :
%d\nEksemplar    : %d\nRating      : %d\n",
                buku.judul, buku.penulis, buku.penerbit,
buku.tahun, buku.eksemplar, buku.rating)
            return
        } else if pustaka.pustaka[mid].rating < r {
            high = mid - 1
        } else {
            low = mid + 1
        }
    }
    fmt.Println("Tidak ada buku dengan rating seperti
itu.")
}

func main() {
    var pustaka DaftarBuku
    var n int

```

```

    fmt.Print("Masukkan jumlah buku: ")
    fmt.Scanln(&n)

    for i := 0; i < n; i++ {
        var id, eksemplar, tahun, rating int
        var judul, penulis, penerbit string

        fmt.Printf("Masukkan data buku ke-%d:\n", i+1)
        fmt.Print("ID          : ")
        fmt.Scanln(&id)
        fmt.Print("Judul        : ")
        fmt.Scanln(&judul)
        fmt.Print("Penulis       : ")
        fmt.Scanln(&penulis)
        fmt.Print("Penerbit      : ")
        fmt.Scanln(&penerbit)
        fmt.Print("Eksemplar     : ")
        fmt.Scanln(&eksemplar)
        fmt.Print("Tahun         : ")
        fmt.Scanln(&tahun)
        fmt.Print("Rating        : ")
        fmt.Scanln(&rating)

        buku := Buku{id, judul, penulis, penerbit,
eksemplar, tahun, rating}
        DaftarkanBuku(&pustaka, buku)
        fmt.Println()
    }

    CetakTerfavorit(pustaka, n)
    UrutBuku(&pustaka, n)
    Cetak5Terbaru(pustaka, n)
    var targetRating int
    fmt.Print("\nMasukkan rating buku yang ingin dicari:
")
    fmt.Scanln(&targetRating)
    CariBuku(pustaka, n, targetRating)

```

```
}
```

Screenshot

```
Tahun      : 2012
Rating     : 5

5 Buku Dengan Rating Tertinggi:
Judul      : Manusia
Rating     : 5

Masukkan rating buku yang ingin dicari: 5
Buku ditemukan:
Judul      : Manusia
Penulis    : angit
Penerbit   : azka
Tahun      : 2012
Eksemplar  : 1000
Rating     : 5
PS C:\2311102230_M. AZKA HERMAWAN_MODUL-12> |
```

Deskripsi:

Program ini adalah aplikasi manajemen data buku yang memungkinkan pengguna untuk menginput data buku, melihat buku dengan rating tertinggi, mengurutkan buku berdasarkan rating, serta melakukan pencarian buku berdasarkan rating tertentu.