LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL 8

PENCARIAN NILAI ACAK PADA HIMPUNAN DATA

Oleh:

Achmad Zulvan Nur Hakim

103112400070

IF-12-01

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

I. DASAR TEORI

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..

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 $\boldsymbol{kr}$ s.d. kanan $\boldsymbol{kn}$. 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.

II. GUIDED

Code 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]=%.f\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 stop %d: cek arr[%d]= %.f\n", iterations, mid,
arr[mid])

		if arr[mid] == target {
			return mid, iterations
		} else if target < arr[mid] {
			high = mid - 1
		} else {
			low = mid - 1
		}
	}
	return -1, iterations
}
func main() {

	data := []float64{2, 7, 9, 1, 5, 6, 18, 13, 25, 20}
	target := 13.0
```

```
	fmt.Println("sequenyial search (data tidak perlu urut):")
	idxseq, iterseq := sequentialSearch(data, target)
	if idxseq != -1 {
		fmt.Printf("hasil : ditemukan di indeks %d dalam %d
langkah\n\n", idxseq, iterseq)
	} else {
		fmt.Printf("hasil : tidak ditemukan setelah %d langkah\n",
iterseq)
	}

	sort.Float64s(data)
	fmt.Println("Binary search (setelah data diurutkan):")
	fmt.Println("Data terurut:", data)

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

Output:

```
PS D:\103112400070> go run "d:\103112400070\103112400070_Modul8\Guided\1\1.go"
sequenyial search (data tidak perlu urut):
sequential step 1: cek arr [0]=2
sequential step 2: cek arr [1]=7
sequential step 3: cek arr [2]=9
sequential step 4: cek arr [3]=1
sequential step 5: cek arr [4]=5
sequential step 6: cek arr [5]=6
sequential step 7: cek arr [6]=18
sequential step 8: cek arr [7]=13
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 stop 1: cek arr[4]= 7
binary stop 2: cek arr[6]= 13
Hasil : Ditemukan indeks 6 dalam 2 langkah
```

Penjelasan:

Program ini membandingkan efisiensi sequential search dan binary search dalam mencari angka tertentu pada array. Sequential search mencari satu per satu, sedangkan binary search yang membutuhkan data terurut lebih cepat karena membagi ruang pencarian setiap langkahnya.

Code 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:

```
PS D:\103112400070> go run "d:\103112400070\103112400070_Modul8\Guided\2\2.go"
Sequential Search - Cari nama 'Fajar'
Ditemukan di indeks: 5, Iterasi: 6

Binary Search - Cari NIM '2206'
Ditemukan di indeks: 5, Iterasi: 3
```

Penjelasan:

Program ini mencari data mahasiswa menggunakan sequential search untuk nama dan binary search untuk NIM. Data mahasiswa disimpan dalam array dan berisi nama, NIM, kelas, jurusan, dan IPK. Setelah data diurutkan berdasarkan NIM, hasil pencarian menampilkan indeks dan jumlah iterasi dari masing-masing metode.

III. UNGUIDED

Code 1:

```
// Achmad Zulvan Nur Hakim 103112400070
package main

import "fmt"

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

func prosesSuara(data []int) (PilihanKaRT, int) {
	p := PilihanKaRT{
		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 main() {
	var input int
	var data []int

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

	hasilPungutan, iterasi := prosesSuara(data)

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

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

Output:

```
PS D:\103112400070> go run "d:\103112400070\103112400070_Modul8\Unguided\1\1.go"
7 19 3 2 78 3 1 -3 18 19 0

Total iterasi: 11
Suara masuk: 10
Suara sah: 8
Hasil:
1: 1 suara
2: 1 suara
3: 2 suara
7: 1 suara
18: 1 suara
19: 2 suara
```

Penjelasan:

Program ini mencatat dan menghitung hasil pemilihan ketua RT dengan maksimal 20 kandidat. Data suara dimasukkan hingga nilai 0 sebagai penanda akhir input, lalu diproses untuk menghitung total suara, suara sah, dan distribusi suara tiap kandidat. Hasilnya ditampilkan dalam bentuk jumlah iterasi input, total suara sah, dan perolehan suara masing-masing kandidat yang mendapat suara.

Code 2:

```
// Achmad Zulvan Nur Hakim 103112400070
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:

Penjelasan:

Program ini memproses pemungutan suara untuk memilih ketua dan wakil ketua RT dari maksimal 20 calon. Setelah semua suara dimasukkan, program menghitung jumlah suara sah dan menentukan dua calon dengan suara terbanyak. Calon dengan suara terbanyak menjadi ketua, sedangkan yang kedua menjadi wakil ketua.

Code 3:

```
//Achmad Zulvan Nur Hakim 103112400070
package main

import "fmt"

const MAX = 1000000
var data [MAX]int

func main() {
    var n, k int
    fmt.Scan(&n, &k)

    isiArray(n)
    pos := posisi(n, k)
    if pos == -1 {
        fmt.Println("TIDAK ADA")
    } else {
        fmt.Println(pos)
    }
}

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

func posisi(n, k int) int {
    left := 0
    right := n - 1

    for left <= right {
        mid := left + (right-left)/2

        if data[mid] == k {
            return mid
        }

        if data[mid] < k {
            left = mid + 1
        } else {
            right = mid - 1
```

```
        }
    }

    return -1
}
```

Output:

```
PS D:\103112400070> go run "d:\103112400070\103112400070_Modul8\Unguided\3\3.go"
12 534
1 3 8 16 32 123 323 323 534 543 823 999
8
PS D:\103112400070> go run "d:\103112400070\103112400070_Modul8\Unguided\3\3.go"
12 535
1 3 8 16 32 123 323 323 534 543 823 999
TIDAK ADA
```

Penjeasan:

Program ini mencari posisi suatu bilangan *k* dalam array *data* menggunakan algoritma binary search. Pertama, program mengisi array sebanyak *n* elemen, kemudian mencari nilai *k* pada array yang diasumsikan sudah terurut. Jika ditemukan, posisi indeksnya dicetak. Jika tidak, program menampilkan "TIDAK ADA".

IV. KESIMPULAN

V. REFERENSI

MODUL 8. PENCARIAN NILAI ACAK PADA HIMPUNAN DATA