Searching

Tim Ajar Struktur Data
Genap 2022/2023

Capaian Pembelajaran

- Mahasiswa memahami algoritma *linear search* pada Array
- Mahasiswa memahami penerapan algoritma linear search pada Array of object

Definisi Searching

- Searching adalah proses untuk menemukan suatu data atau informasi dari sekumpulan data/informasi yang ada.
- Algoritma pencarian merupakan algoritma yang menerima suatu kata kunci sebagai kriteria pencarian, dan dengan langkah-langkah tertentu akan mencari data yang sesuai dengan kata kunci tersebut.

Hasil Searching

- Hasil atau keluaran dari proses pencarian dapat berupa:
- Pesan
  - Ditemukan/ Ada
  - Tidak ditemukan/ Tidak ada
- Index array
  - index = 13
  - i = 7
  - idx = -1 (jika data yang dicari tidak ditemukan)
- Nilai Boolean
  - TRUE
  - FALSE

Sequential Search

- Sequential Search atau disebut juga Linear Search adalah teknik pencarian data dimana data dicari secara urut dari depan ke belakang atau dari awal sampai akhir
- Proses pencarian dilakukan dengan membandingkan elemen array satu per satu secara beruntun mulai dari elemen pertama sampai elemen yang dicari sudah ditemukan atau sampai semua elemen sudah diperiksa

Kelebihan Sequential Search

- Kumpulan data tidak harus dalam keadaan terurut
- Jika data yang dicari terletak di posisi depan, maka data akan ditemukan dengan cepat
- Penyisipan dan penghapusan elemen pada kumpulan data tidak mempengaruhi proses pencarian karena data tidak perlu diurutkan.
  Pada algoritma pencarian lainnya, data harus disusun kembali setelah adanya penyisipan atau penghapusan elemen
- Merupakan algoritma pencarian yang sangat sederhana, hemat sumber daya dan memori

Kekurangan Sequential Search

- Jika data yang dicari terletak di posisi belakang atau paling akhir, maka proses pencarian akan membutuhkan waktu yang lama
- Beban komputer akan semakin bertambah jika jumlah data dalam array sangat banyak, sehingga tidak cocok untuk data berukuran besar

Algoritma Sequential Search

Secara umum, algoritma Sequential Search dijabarkan sebagai berikut:

- Input x (data yang dicari)
- Bandingkan x dengan data ke-i sampai n (n → jumlah elemen array)
- Jika ada data yang sama dengan x maka cetak pesan “ditemukan”
- Jika tidak ada data yang sama dengan x cetak pesan “tidak ditemukan”

Jenis Sequential Search

Berdasarkan urutan datanya, sequential search dibedakan menjadi:

- Unordered Sequential Search
- Ordered Sequential Search

Jenis Sequential Search

Unordered Sequential Search

- Data berada pada keadaan acak, tidak terurut
- pencarian harus dilakukan mulai dari indeks awal sampai indeks terakhir dari data, atau pencarian berhenti ketika data sudah ditemukan

Ordered Sequential Search

- Data sudah dalam keadaan terurut
- Pencarian dilakukan bersamaan dengan perbandingan nilai keyword dengan data yang dikunjungi
- Dapat mengurangi waktu komputasi pencarian

Jenis Sequential Search (Ilustrasi)

Unordered Sequential Search

- Sekumpulan kertas hasil ujian belum diurutkan secara alfabet. Jika seorang siswa ingin mengetahui nilai ujiannya, maka dia harus mencari semua hasil ujian satu persatu sampai kertas hasil ujiannya ditemukan

Ordered Sequential Search

- Jika kita mencari nomor *handphone* “Patrick” di daftar kontak kita, maka kita tidak perlu mencari nama selain “P” untuk menentukan bahwa kita menyimpan nomor handphone Patrick atau tidak

Unordered Sequential Search

- **Input**: array **A**, banyaknya elemen **n**, nilai/kata kunci yang akan dicari **x**
- **Output**: jika ditemukan, tampilkan posisi index **i**
  jika tidak ditemukan, tampilkan pesan **“x tidak ditemukan”**
- **Algoritma**:
  1. Bandingkan x dengan setiap elemen di array A[i] mulai dari indeks 0
  2. Jika x = A[i], hentikan pencarian dan tampilkan posisi index i
  3. Jika tidak, tetap lakukan pencarian untuk elemen selanjutnya sampai dengan index terakhir dari array A
  4. Jika elemen tidak ditemukan di dalam array A, tampilkan "x tidak ditemukan"

Unordered Sequential Search

- Contoh 1:

Index	0	1	2	3	4	5	6	7
Array A	34	16	25	33	7	29	48	14

- Jika nilai yang dicari x = 33 pada array A, maka kita akan membandingkan x dengan **34, 16, 25, dan 33** masing-masing sebanyak satu kali
- Kita menemukan 33 pada posisi **index = 3** sebagai hasil keluaran
- Banyaknya perbandingan yang dilakukan adalah **4 kali**

Unordered Sequential Search

- Contoh 2:

Index	0	1	2	3	4	5	6	7
Array A	34	16	25	33	7	29	48	14

- Jika nilai yang dicari x = 18 pada array A, maka kita akan membandingkan x dengan **34, 16, 25, 33, 7, 29, 48, dan 14** masing-masing sebanyak satu kali
- Setelah mencari di semua elemen array, kita tidak menemukan 18, sehingga hasil keluaran berupa **“18 tidak ditemukan”**
- Banyaknya perbandingan yang dilakukan adalah **8 kali**

Unordered Sequential Search

- Pada unordered sequential search, jika kita ingin mencari x di dalam array yang tidak berurutan dengan jumlah elemen sebanyak n, maka terdapat kemungkinan terjadi kasus terburuk (worst case)
- **Worst case** terjadi jika kita harus **melakukan pencarian pada seluruh elemen array** untuk mendapatkan data yang dicari.
- Hal ini berarti kita harus melakukan proses perbandingan sebanyak **n kali**

Ordered Sequential Search

- **Input**: array **A**, banyaknya elemen **n**, nilai/kata kunci yang akan dicari **x**
- **Output**: jika ditemukan, tampilkan posisi index **i**
  jika tidak ditemukan, tampilkan pesan **“x tidak ditemukan”**
- **Algoritma**:
  - Bandingkan x dengan setiap elemen di array A[i] mulai dari indeks 0
  - Jika x = A[i], hentikan pencarian dan tampilkan posisi index i
  - Jika tidak, bandingkan x dengan elemen itu lagi dan cek apakah x LEBIH BESAR dari A[i]
  - Jika x > A[i], tetap lakukan pencarian untuk elemen selanjutnya
  - Jika tidak, artinya x < A[i], hentikan pencarian dan tampilkan "x tidak ditemukan"

Ordered Sequential Search

- Contoh 1:

Index	0	1	2	3	4	5	6	7
Array A	7	14	16	25	29	33	34	48

- Jika nilai yang dicari x = 33 pada array A, maka kita akan membandingkan x dengan 7, 14, 16, 25, dan 29 masing-masing sebanyak dua kali (untuk “=“ dan untuk “>”),
  kemudian membandingkan x dengan 33 sebanyak satu kali (untuk “=“)
- Kita menemukan 33 pada posisi index = 5 sebagai hasil keluaran
- Banyaknya perbandingan yang dilakukan adalah $2 \times 5 + 1 = 11$ kali

Ordered Sequential Search

- Contoh 2:

Index	0	1	2	3	4	5	6	7
Array A	7	14	16	25	29	33	34	48

- Jika nilai yang dicari x = 18 pada array A, maka kita akan membandingkan x dengan 7, 14, 16, dan 25 masing-masing sebanyak dua kali (untuk “=“ dan untuk “>”).
  Pada perbandingan terakhir (jika x > 25), kita mendapat jawaban "TIDAK“. Artinya semua elemen setelah 25 di dalam array ini lebih besar dari x
- Dengan demikian, hasil keluaran berupa “18 tidak ditemukan”
- Banyaknya perbandingan yang dilakukan adalah $2 \times 4 = 8$ kali

Ordered Sequential Search

- Pada ordered sequential search, jika kita ingin mencari x di dalam array yang berurutan dengan jumlah elemen sebanyak n, maka terdapat kemungkinan terjadi kasus terburuk (worst case)
- *Worst case* terjadi jika nilai x lebih besar dari elemen terakhir di dalam array, sehingga kita harus melakukan pencarian pada seluruh elemen array untuk mendapatkan data yang dicari.
- Hal ini berarti kita harus melakukan proses perbandingan sebanyak $n \times 2$ kali, sebuah n untuk perbandingan “=“ dan n yang lain untuk perbandingan “>”

Algoritma Pencarian (Return Index Array)

```
for (i = 0; i < n; i++) {
   if (x == data[i]) {
      return (i);
   }
}
return (-1);
```

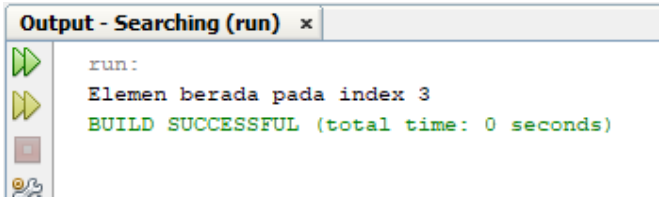
Membandingkan data yang dicari (x)
dengan data pada array index ke-i

Mengembalikan nilai index ke-i
jika data ditemukan

Mengembalikan nilai -1
jika data tidak ditemukan

Contoh Program (Prosedural)

```
public static int search(int arr[], int x) {
    int n = arr.length;
    for (int i = 0; i < n; i++) {
        if (arr[i] == x) {
            return i;
        }
    }
    return -1;
}

public static void main(String args[]) {
    int arr[] = {34, 16, 25, 33, 7, 29, 48, 14};
    int x = 33;

    int hasil = search(arr, x);
    if (hasil == -1) {
        System.out.println("Elemen yang dicari tidak ada di dalam array");
    } else {
        System.out.println("Elemen berada pada index " + hasil);
    }
}
```



Algoritma Pencarian (Return Boolean)

```
for (i = 0; i < n; i++) {
   if (x == data[i]) {
      return TRUE;
   }
}
return FALSE;
```

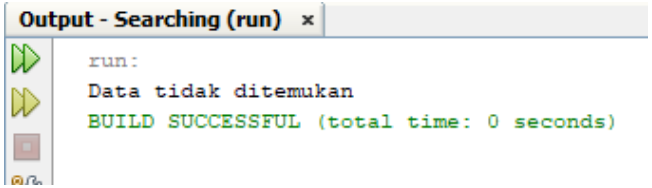
Membandingkan data yang dicari (x) dengan data pada array index ke-i

Mengembalikan nilai TRUE jika data ditemukan

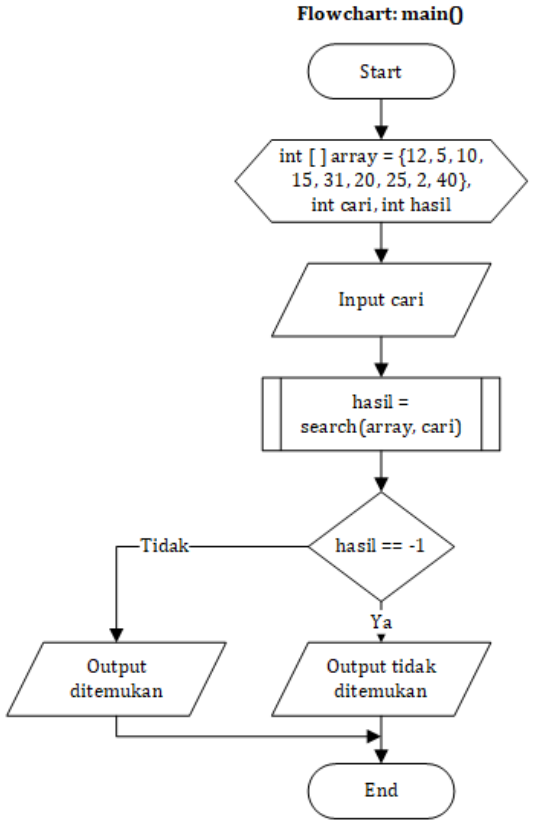
Mengembalikan nilai FALSE jika data tidak ditemukan

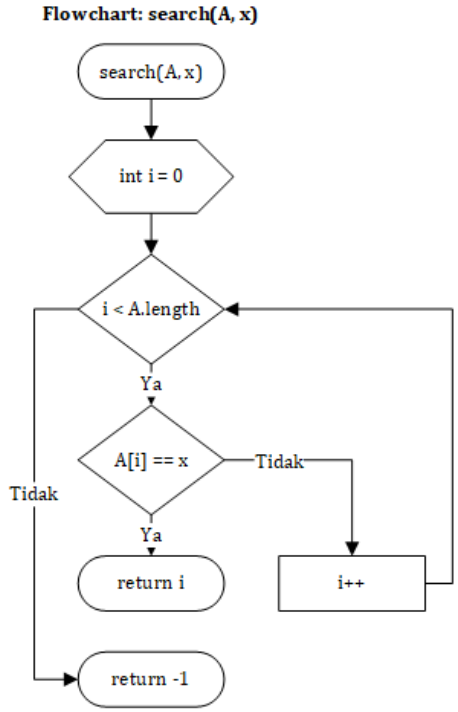
Contoh Program (Prosedural)

```java
public static boolean search(int[] arr, int x) {
    for (int i = 0; i < arr.length; i++) {
        if (arr[i] == x) {
            return true;
        }
    }
    return false;
}

public static void main(String a[]) {

    int[] arr = {12, 5, 10, 15, 31, 20, 25, 2, 40};
    int x = 7;

    boolean ditemukan = search(arr, x);
    if (ditemukan) {
        System.out.println("Data ditemukan");
    } else {
        System.out.println("Data tidak ditemukan");
    }
}
```



Flowchart Sequential Search (Prosedural)





Sequential Search pada ARRAY of OBJECT

Sequential Search pada Array of Object

- Pencarian menggunakan Sequential Search pada array of object mempunyai cara yang sama seperti pencarian pada array biasa
- Langkah-langkah:
  - Class, atribut, dan method harus dibuat terlebih dahulu
  - Melakukan instansiasi objek pada fungsi Main di Class lain
  - Menambahkan proses pencarian pada fungsi Main untuk mencari sebuah data
  - Data yang dicari biasanya berkaitan dengan atribut

Contoh Program

- Menggunakan object
  (modifikasi dari contoh Prosedural)

```java
public class LinearSearchMain {

    public static void main(String args[]) {
        int arr[] = {34, 16, 25, 33, 7, 29, 48, 14};
        LinearSearch ls = new LinearSearch(arr);
        int x = 33;

        int hasil = ls.search(arr, x);
        if (hasil == -1) {
            System.out.println("Elemen yang dicari tidak ada di dalam array");
        } else {
            System.out.println("Elemen berada pada index " + hasil);
        }
    }
}
```

```java
public class LinearSearch {
    public int[] data;

    public LinearSearch(int[] arr) {
        data = new int[arr.length];
        for (int i = 0; i < arr.length; i++) {
            data[i] = arr[i];
        }
    }

    public int search(int arr[], int x) {
        int n = arr.length;
        for (int i = 0; i < n; i++) {
            if (arr[i] == x) {
                return i;
            }
        }
        return -1;
    }
}
```

Contoh

Misalkan terdapat class Mahasiswa dengan atribut nim dan nama, serta konstruktor

```
public class Mahasiswa {
public String nim;
public String nama;

        public Mahasiswa(String ni, String na){
                nim = ni;
                nama = na;
        }
}
```

Contoh (Lanjutan)

Pada class lain, misalnya MahasiswaMain, dilakukan deklarasi objek array dengan ukuran 10

```
public class MahasiswaMain {
    public static void main (String[] args){
        Mahasiswa[] mhs = new Mahasiswa[10];
    }
}
```

Contoh (Lanjutan)

Selanjutnya, instansiasi objek dan pengisian nilai atribut dilakukan sebanyak 10 kali

```
mhs[0] = new Mahasiswa(“167829”, “Dendi”);
mhs[1] = new Mahasiswa(“167173”, “Safira”);
mhs[2] = new Mahasiswa(“167442”, “Yuri”);
...
mhs[9] = new Mahasiswa(“167301”, “Lita”);
```

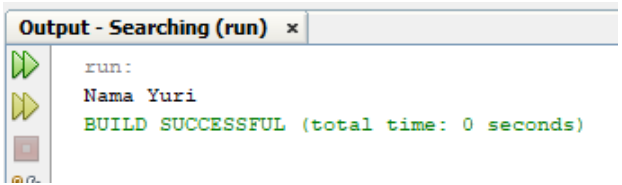
Contoh (Lanjutan)

Setelah itu, perulangan dan percabangan ditambahkan untuk proses pencarian. Misalkan menampilkan nama mahasiswa berdasarkan hasil pencarian nim

```
String cari = "167442";
for(int i=0; i<10; i++){
    if(cari.equals(mhs[i].nim)){
        System.out.println ("Nama " + mhs[i].nama);
    }
}
```

Contoh Program

```java
public class Mahasiswa {
    public String nim;
    public String nama;

    public Mahasiswa(String ni, String na) {
        nim = ni;
        nama = na;
    }
}
```




```java
public class MahasiswaMain {
    public static void main(String[] args) {
        Mahasiswa[] mhs = new Mahasiswa[10];
        mhs[0] = new Mahasiswa("167829", "Dendi");
        mhs[1] = new Mahasiswa("167173", "Safira");
        mhs[2] = new Mahasiswa("167442", "Yuri");
        mhs[3] = new Mahasiswa("167453", "Deva");
        mhs[4] = new Mahasiswa("167712", "Candra");
        mhs[5] = new Mahasiswa("167146", "Roni");
        mhs[6] = new Mahasiswa("167276", "Dewi");
        mhs[7] = new Mahasiswa("167330", "Latif");
        mhs[8] = new Mahasiswa("167515", "Putri");
        mhs[9] = new Mahasiswa("167301", "Lita");
        String cari = "167442";
        for (int i = 0; i < mhs.length; i++) {
            if (cari.equals(mhs[i].nim)) {
                System.out.println("Nama " + mhs[i].nama);
            }
        }
    }
}
```

Latihan 1

Diketahui array di bawah ini:

Index	0	1	2	3	4	5	6
Array	46	7	3	17	30	25	20

- Jelaskan algoritma untuk mencari bilangan x pada array tersebut dengan menggunakan
  - Ordered Sequential Search
  - Unordered Sequential Search
- Berapa kali perbandingan dilakukan?
- Jenis Sequential Search manakah yang cocok untuk digunakan pada array di atas? Mengapa?

*Dengan nilai x:

a. x=20

b. x=21

Latihan 2

Diketahui array di bawah ini:

Index	0	1	2	3	4	5	6
Array	3	7	17	20	25	30	46

- Jelaskan algoritma untuk mencari bilangan x pada array tersebut dengan menggunakan
  - Ordered Sequential Search
  - Unordered Sequential Search
- Berapa kali perbandingan dilakukan?
- Jenis Sequential Search manakah yang cocok untuk digunakan pada array di atas? Mengapa?

*Dengan nilai x:

a. x=20

b. x=21

Latihan 3

Terdapat array A satu dimensi yang dibuat dengan int A[10], sudah berisi bilangan sebagai berikut:

index	0	1	2	3	4	5	6	7	8	9
elemen	13	5	2	9	10	6	14	4	3	12

Susun flowchart untuk menginput sebuah nilai integer (misal x), kemudian periksa isi array, apakah ada isi array yang nilainya sama dengan x. Bila ada, cetak string “ADA” dan posisi (nomor indeks)-nya pada array A, bila tidak ada cetak string “TIDAK ADA”.