[3] Sequential

- ✧ **Algoritma Sequential**: merupakan struktur algoritma paling dasar yang berisi rangkaian instruksi yang diproses secara satu per satu, mulai dari instruksi pertama sampai instruksi terakhir.
- ✧ Contoh algoritma sequential adalah algoritma untuk menjumlahkan dua angka, mencetak daftar, atau mengurutkan data dalam bentuk urutan tertentu secara linear.

```
public class SequentialExample {
    public static void main(String[] args) {
        // Langkah 1: Mendeklarasikan dua angka
        int a = 3;
        int b = 5;

        // Langkah 2: Menjumlahkan kedua angka
        int sum = a + b;

        // Langkah 3: Mencetak hasil
        System.out.println("Hasil: " + sum);
        // Output: Hasil: 8
    }
}
```

- ✧ **Struktur Data Sequential**: Struktur data yang menyimpan elemen-elemen dalam urutan tertentu, dan akses terhadap elemen-elemen tersebut biasanya dilakukan satu per satu dalam urutan tersebut.
- ✧ Contoh struktur data sequential adalah array, daftar (list), queue (antrian), dan stack (tumpukan). Akses ke elemen-elemen dalam struktur ini umumnya terjadi secara berurutan dari awal hingga akhir, atau sebaliknya.

```
public class SequentialArrayExample {
    public static void main(String[] args) {
        // Membuat array (struktur data sequential)
        int[] numbers = {1, 2, 3, 4, 5};

        // Mengakses elemen-elemen array secara sequential
        for (int i = 0; i < numbers.length; i++) {
            System.out.println(numbers[i]);
        }
    }
}
```

SequentialSearch

- ✧ **Searching** adalah proses menemukan elemen tertentu di dalam kumpulan data. Misalnya, mencari angka dalam array atau daftar yang berisi elemen-elemen.
- ✧ **Sequential Search** (atau Linear Search) adalah metode pencarian di mana setiap elemen dalam kumpulan data diperiksa satu per satu secara berurutan, mulai dari elemen pertama hingga elemen terakhir, sampai elemen yang dicari ditemukan atau seluruh data telah diperiksa.

```
public class SequentialSearch {
    public static void main(String[] args) {
        int[] data = {3, 5, 7, 9, 11};
        int target = 7;

        // Sequential search
        boolean found = false;
        for (int i = 0; i < data.length; i++) {
            if (data[i] == target) {
                found = true;
                break;
            }
        }

        if (found) {
            System.out.println("Elemen ditemukan.");
        } else {
            System.out.println("Elemen tidak ditemukan.");
        }
    }
}
```

Kelebihan

- ✧ Sederhana dan Mudah Diimplementasikan.
- ✧ Kinerja Konstan pada Data Kecil.

Kekurangan

- ✧ Tidak Efisien pada Data Besar. Karena setiap elemen harus diperiksa satu-persatu.
- ✧ Tidak Memanfaatkan Pengurutan. Karena tetap memeriksa setiap elemen.
- ✧ Tidak Optimal untuk Data yang Terstruktur, seperti hashmap dan binary tree.

Kesimpulan

- ✧ Sequential search cocok untuk data kecil dan tidak terurut karena kesederhanaannya.

Pengurutan (Sorting)

- ✧ **Definisi**: Pengurutan adalah proses mengatur sekumpulan objek berdasarkan susunan tertentu
- ✧ Semua algoritma sorting selalu melakukan operasi perbandingan elemen untuk menemukan posisi urutan yang tepat

Algortima Sorting

Banyak algoritma sorting telah ditemukan, menunjukkan bahwa masalah pengurutan memiliki beragam solusi. Beberapa algoritma sorting yang dikenal antara lain: bubble sort, selection sort, insertion sort, heap sort, shell sort, quick sort, merge sort, dan sebagainya.

Bubble Sort

- ✧ **Inspirasi:** Algoritma bubble sort terinspirasi oleh gelembung sabun yang mengapung di permukaan air. Zat yang lebih ringan (gelembung) akan terapung, sementara yang lebih berat (batu) akan tenggelam**.**
- ✧ **Prinsip Kerja**: Dalam bubble sort, jika kita ingin mengurutkan array secara ascending (menaik), elemen dengan nilai terkecil akan "diapungkan" ke atas melalui proses pertukaran elemen.
- ✧ **Proses**: Bubble sort menggunakan prinsip pertukaran (exchange) elemen selama proses pengurutan.

Berikut ini contoh program penggunaan Bubble Sort secara ascending:

```
public class BubbleSort {
   public static void main(String[] args) {
      int array[] = {9, 1, 8, 2, 7, 3, 6, 4, 5};
      bubbleSort(array);
      for (int i : array) {
         System.out.print(i);
      }
   }
   public static void bubbleSort(int array[]) {
      for (int i = 0; i < array.length - 1; i++) {
         for (int j = 0; j < array.length - i - 1; j++) {
            if (array[j] > array[j + 1]) {
               int temp = array[j];
               array[j] = array[j + 1];
               array[j + 1] = temp;
            }
         }
      }
   }
}
```

Kelebihan

- ✧ Sederhana dan Mudah Dipahami.
- ✧ Baik untuk Data Kecil.

Kekurangan

- ✧ Tidak efisien, karena banyak operasi pertukaran yang dilakukan setiap langkah pengapungan
- ✧ Untuk ukuran array yang besar, algoritma ini membutuhkan waktu yang lama

Kesimpulan

Bubble Sort lebih cocok untuk situasi di mana kesederhanaan dan kemudahan implementasi lebih penting daripada efisiensi, seperti dalam pembelajaran atau pengurutan dataset kecil.