Praktikum Day #1

Correctness of Algorithm

Lab Assistant :
Fikri & Jo

Overview

- Analyzing algorithm
- Correctness of algorithm
- Method used
- Exercize

Mengapa harus menganalisis algoritma ?

EFISIENSI

- Minimasi Waktu Eksekusi (Time Complexity).
- Minimasi Penggunaan Memori (Space Complexity).
- Dapat Diskalakan (Scalability)

EFEKTIFITAS

- Algoritma harus benar dalam menyelesaikan masalah.
- memilih algoritma yang paling optimal untuk suatu masalah
- Konsisten dalam menghasilkan output (yang benar).

Apa itu correctness of algorithm?

- **Partial Correctness** :
  Algoritma memberikan hasil yang benar **jika** ia berhenti**.**
- **Termination (berhenti)** **:**
  Algoritma **pasti** berhenti untuk setiap input yang benar.
- **Total Correctness** **:**
  Algoritma **pasti** berhenti untuk setiap input yang benar.

Case :

Seorang programmer ingin mengimplementasikan algoritma untuk menghitung **jumlah deret aritmetika dari 1 hingga *n*** menggunakan iterasi.

```java
public class Test{
    public static int sum(int n) {
        int sum = 1;
        for (int i = 2; i <= n; i++) {
            sum += i;
        }
        return sum;
    }
```

Metode dalam Analisis Algoritma

- Induksi Matematika
- Loop Invariant
- Metode Formal

Induksi Matematika

Step :

- **Basis induksi :** Buktikan bahwa algoritma benar untuk nilai terkecil (***n=1***)**.**
- **Hipotesis Induksi :** Asumsikan algoritma benar untuk suatu nilai **(n = k)**
- **Langkah Induksi :** Buktikan algoritma benar untuk suatu nilai **(n = k+1)**

CONTOH

Buktikan Algoritma berikut dengan Induksi Matematika

```
public static int sum(int n){
    return (n * (n + 1)) / 2;
}
```

SOLUSI

$$S(n) = 1 + 2 + 3 + \ldots + n = \frac{n(n+1)}{2}$$

Step 1 : Basis Induksi

Untuk $n = 1$:

$$S(1) = \frac{1(1+1)}{2} = \frac{2}{2} = 1$$

Step 2 : Hipotesis Induksi

$n = k$

$$S(k) = 1 + 2 + 3 + \ldots + k = \frac{k(k+1)}{2}$$

Step 3 : Langkah Induksi

$$S(k+1) = S(k) + (k+1)$$

$$S(k+1) = \frac{k(k+1)}{2} + (k+1)$$

$$= \frac{k(k+1) + 2(k+1)}{2}$$

$$= \frac{(k+1)(k+2)}{2}$$

Terbukti

CONTOH 2

Buktikan Algoritma berikut dengan Induksi Matematika

```java
void sumEven(int n) {
    int sum = 0;
    for (int i = 2; i <= n; i += 2) {
        sum += i;
    }
    System.out.println("Jumlah bilangan genap hingga " + n + " adalah: " + sum);
}
```

SOLUSI

$$S(n) = 2 + 4 + 6 + \ldots + 2n = \frac{2n(2n+2)}{4}$$

Step 1 : Basis Induksi

Untuk $n = 1$

$$S(1) = \frac{2(1)(2(1)+2))}{4} = 2$$

Step 2 : Hipotesis Induksi

$n = k$

$$S(k) = 2 + 4 + 6 + \ldots + 2k = \frac{2k(2k+2)}{4}$$

Step 3 : Langkah Induksi

Substitusi $(k+1)$ ke dalam Hipotesis

Kita ingin membuktikan : $S(k+1) = \frac{4k^2+12k+8}{4}$

$$\begin{aligned} S(k+1) &= S(k) + 2(k+1) \\ &= \frac{2k(2k+2)}{4} + (2k+2) \\ &= \frac{4k^2+4k}{4} + \frac{8k+8}{4} \\ &= \frac{4k^2+12k+8}{4} \end{aligned}$$

Terbukti

LOOP INVARIANT

Step :

- **Inisialisasi** → Buktikan bahwa loop invariant benar sebelum loop mulai berjalan.
- **Maintenance** -> Buktikan bahwa jika loop invariant benar di suatu iterasi, maka tetap benar di iterasi berikutnya.
- **Terminasi** -> Buktikan bahwa ketika loop berhenti, loop invariant memberikan hasil yang benar sesuai spesifikasi algoritma.

CONTOH

Buktikan Algoritma berikut dengan Loop Invariant

```
void faktorial(int n) {
    int hasil = 1;
    for (int i = 1; i <= n; i++) {
        hasil *= i;
    }
    System.out.println("Faktorial dari " + n + " adalah: " + hasil);
}
```

SOLUSI

$$n! = n \times (n-1) \times \ldots \times 1.$$

Inisialisasi

Sebelum loop dimulai, nilai awal adalah:

- $hasil = 1$
- $i = 1$

Pada saat ini, loop invariant:

$hasil = 1! = 1$

Maintenance

Misalkan loop invariant benar pada iterasi ke $-\ k$, *yaitu:*

$hasil = k!$

pada iterasi berikutnya $(1 = k + 1)$
kita mengalikan dengan $k + 1$
$hasil = k!\ \times (k + 1) = (k + 1)!$

SOLUSI

Terminasi

Loop berhenti saat $i = n$

artinya semua bilangan telah dikalikan dari 1 hingga n.

hasil = n!

Terbukti

METODE FORMAL

Pembuktian formal digunakan untuk menunjukkan bahwa suatu algoritma benar dengan menggunakan logika matematis dan notasi formal. Kunci dari pembuktian ini ada pada:

- **Preconditions** → Keadaan yang harus benar **sebelum** algoritma dijalankan.
- **Postconditions** → Keadaan yang harus benar **setelah** algoritma selesai dijalankan.

CONTOH

Buktikan Algoritma berikut dengan Metode Formal

```
    public static void main(String[] args) {
        int number = 6;
        if (isPrime(n: number)) {
            System.out.println(number + " adalah bilangan prima.");
        } else {
            System.out.println(number + " bukan bilangan prima.");
        }
    }
public static boolean isPrime(int n) {
        if (n <= 1) {
            return false; //
        }
        for (int i = 2; i * i <= n; i++) {
            if (n % i == 0) {
                return false;
            }
        }
        return true;
    }
```

SOLUSI

Spesifikasi Algoritma

- **Precondition**:
  - Input n adalah bilangan bulat positif ($n > 1$).
- **Postcondition**:
  - Algoritma mengembalikan `true` jika n adalah bilangan prima, dan `false` jika bukan.
  - Definisi bilangan prima: n adalah bilangan prima jika dan hanya jika n hanya habis dibagi oleh 1 dan dirinya sendiri.

SOLUSI

Base Case:

- **Kasus**: $n = 2$.
- **Precondition**: $n = 2$ (bilangan bulat positif).
- **Postcondition**: Algoritma harus mengembalikan `true` karena 2 adalah bilangan prima.
- **Pembuktian**:
  - Algoritma memeriksa apakah n habis dibagi oleh bilangan dari 2 hingga $\sqrt{2} \approx 1.414$.
  - Karena tidak ada bilangan bulat antara 2 dan 1.414, algoritma langsung mengembalikan `true`.
  - **Kesimpulan**: Postcondition terpenuhi.

SOLUSI

Induksi n>2:

- **Asumsi Induktif**:
  - Algoritma benar untuk semua bilangan bulat k dimana $2 \leq k < n$.
- **Langkah Induktif**:
  - Kita perlu membuktikan bahwa algoritma juga benar untuk n.
  - Algoritma memeriksa semua bilangan i dari 2 hingga $\sqrt{n}$:
    - Jika n habis dibagi oleh i, maka n bukan prima, dan algoritma mengembalikan `false`.
    - Jika tidak ada i yang membagi n, maka n adalah prima, dan algoritma mengembalikan `true`.
  - **Pembuktian**:
    - Jika n bukan prima, maka n memiliki pembagi d dimana $2 \leq d \leq \sqrt{n}$. Algoritma akan menemukan d dan mengembalikan `false`.
    - Jika n adalah prima, maka tidak ada pembagi d yang memenuhi $2 \leq d \leq \sqrt{n}$. Algoritma akan mengembalikan `true`.
  - **Kesimpulan**:
    - Algoritma benar untuk n.

KESIMPULAN

Metode	Fokus	Cakupan	Contoh Algoritma
Induksi Matematika	Membuktikan kebenaran algoritma berbasis pola rekursi atau iterasi menggunakan langkah dasar dan induksi	Algoritma berbasis **rekursi** atau **perulangan** yang memiliki pola matematis	Faktorial, Fibonacci, jumlah deret, Hanoi Tower
Loop Invariant	Membuktikan kebenaran algoritma dengan memastikan kondisi tertentu tetap benar selama perulangan berjalan	Algoritma yang menggunakan **loop (iteratif)**	Pencarian maksimum, Bubble Sort,Insertion Sort, Binary Search
Pembuktian Formal	Membuktikan kebenaran algoritma secara menyeluruh (awal, proses, akhir) dengan precondition dan postcondition	**Semua jenis algoritma**, baik rekursif, iteratif, atau kompleks	Merge Sort, Quick Sort, algoritma Divide and Conquer, algoritma Dynamic Programming

EXERCISE

Menyusul di classroom