ALGORITMA DAN PEMROGRAMAN 2

Algoritma Rekursif

PENDAHULUAN

- Rekursif merupakan kemampuan fungsi/subrutin untuk memanggil dirinya sendiri.
- Fungsi yang memanggil dirinya sendiri disebut fungsi rekursif.

FAKTORIAL

- $n! = n \times (n-1) \times (n-2) \times (n-3) \times \cdots \times 1$
- $n! = n \times (n-1)!$
- $F(n) = \begin{cases} n \times F(n-1) & if\ n > 0 \\ 1 & if\ n = 0 \end{cases}$
- int faktorial(int n){
  if (n == 0)
  return 1;
  else
  return n*faktorial(n-1);
  }

DERET FIBONACCI

- $F(n) = \begin{cases} F(n-1) + F(n-2) & if\ n > 1 \\ n & if\ n = 0,1 \end{cases}$
- 0 1 1 2 3 5 8 13 …

```
Fib(n)
{
   if (n <= 1)
        return n
   F1 ← 0
   F2 ← 1
   for i ← 2 to n
        F ← F1 + F2
        F1 ← F2
        F2 ← F
  return F
}
```

```
Fib(n)
{
    if(n <= 1)
        return n
    else
     return Fib(n-1) + Fib(n-2)
}
```

KOMPLEKSITAS WAKTU ASIMPTOTIK

- Notasi "O" disebut notasi "O-Besar" (Big-O) yang merupakan notasi kompleksitas waktu asimptotik.
- Cara cepat menentukan Big-O:
  - Ambil term yang nilainya paling cepat berkembang
  - Hilangkan koefisiennya
- Misal :

  $T(n) = 2n^2 + 6n + 1$

  Term yg paling cepat berkembang: $2n^2$

  Setelah koefisiennya dibuang maka hasilnya: n^2

  Maka kompleksitas waktunya adalah: $O(n^2)$

RECURSIVE WITH MEMOIZATION

- Membuat algoritma rekursif menjadi efisien
- Jika panggilan rekursif dengan argumen yang sama dilakukan berulang kali, maka algoritma rekursif yang tidak efisien dapat dimoisasi dengan menyimpan nilai yang telah dihitung ke dalam tabel sehingga tidak perlu dihitung ulang.

PSEUDO CODE OF MEMORIZED FIBONACCI ALGORITHM

```
Fib(n)
{
  if n<=1
    return n
  if Fn is in memory
    return Fn
  else
    Fn ← Fib(n-1) + Fib(n-2)
    Save Fn in memory
    return Fn
}
```