

EXERCISE 6: KEBENARAN DAN KOMPLEKSITAS ALGORITMA REKURSIF

untuk dikerjakan sebagai latihan di kelas pada perkuliahan pertemuan ke-6

Petunjuk: Buatlah kelompok beranggotakan 3 orang dan diskusikan permasalahan berikut sebelum pelaksanaan pertemuan. Anda cukup mengerjakan satu soal saja pada setiap bagian, namun tentu saja Anda diharapkan dapat mencoba menganalisis soal yang lainnya.

Bagian 1: Kebenaran algoritma rekursif

1. (Menghitung faktorial)

Algorithm 1 Factorial of a number

```

1: procedure FACTORIAL( $n$ )
2:   if  $n = 1$  then
3:     return 1
4:   else
5:     temp = FACTORIAL( $n - 1$ )
6:     return  $n * \text{temp}$ 
7:   end if
8: end procedure

```

Buktikan kebenaran algoritma rekursif di atas dengan menggunakan **teknik induksi** mengikuti langkah-langkah berikut.

- Buktikan kebenaran algoritma untuk nilai input $n = 1$.
- Berikan hipotesis induksi untuk nilai integer $n \geq 1$.
- Buktikan kebenaran hipotesis Anda. (*Hint: Hipotesis berlaku untuk nilai $n = k$ untuk suatu nilai $k \geq 1$. Pembuktian hipotesis dilakukan dengan mengasumsikan kebenaran algoritma untuk nilai $n = k$, yang mengakibatkan algoritma juga benar untuk nilai $n = k + 1$).*)
- Berikan interpretasi pemahaman Anda terhadap teknik pembuktian ini.

2. (Maksimum array & jumlah array)

Apakah metode pembuktian yang serupa dengan metode di atas dapat digunakan untuk membuktikan kebenaran algoritma berikut?

- Mencari nilai maksimum pada array

Algorithm 2 Finding maximum of an array

```

1: procedure MAX( $A[0..n - 1]$ , int  $n$ )
2:   if  $n = 1$  then return  $A[0]$ 
3:   else
4:      $T = \text{MAX}(A, n - 1)$ 
5:     if  $T < A[n - 1]$  then
6:       return  $A[n - 1]$ 
7:     else
8:       return  $T$ 
9:     end if
10:  end if
11: end procedure

```

b Menghitung jumlah elemen pada array

Algorithm 3 Sum of an array

```
1: procedure SUM( $A[0..n-1]$ , int  $n$ )
2:   if  $n = 1$  then return  $A[0]$ 
3:   else
4:      $S = \text{SUM}(A, n - 1)$ 
5:      $S = S + A[n - 1]$ 
6:     if  $T < A[n - 1]$  then
7:       return  $S$ 
8:     end if
9:   end if
10: end procedure
```

3. (Maksimum array (2))

Buktikan kebenaran algoritma berikut. Apakah teknik pembuktian dengan induksi dapat digunakan dalam hal ini?

Algorithm 4 Finding max of an array

```
1: procedure FINDMAX( $A[i..j]$ ,  $n$ )
2:   if  $n = 1$  then return  $A[0]$ 
3:   end if
4:    $m = \lfloor \frac{i+j}{2} \rfloor$ 
5:    $T_1 = \text{FINDMAX}(A[i..m], \lfloor \frac{n}{2} \rfloor)$ 
6:    $T_2 = \text{FINDMAX}(A[(m+1)..j], n - \lfloor \frac{n}{2} \rfloor)$ 
7:   if  $T_1 \geq T_2$  then return  $T_1$ 
8:   else return  $T_2$ 
9:   end if
10: end procedure
```

▷ i, j are respectively the index of start, end of A

▷ Recursive call the left sub-array

▷ Rec. call right sub-array

▷ Compare the two max elements

Bagian 2: Kompleksitas waktu algoritma rekursif

1. (Maksimum array (2))

Perhatikan kembali Algoritma 4 di atas. Sekarang kita akan menghitung kompleksitas waktu dari algoritma tersebut.

- Misalkan $f(n)$ adalah banyaknya “perbandingan (*comparison*)” yang diterapkan untuk menemukan elemen maksimum dari array dengan ukuran n , dimana $n = 2^k$ untuk suatu bilangan bulat positif k .
- Temukan formula rekursif dari fungsi $f(n)$.

Dalam hal ini, fungsinya adalah sebagai berikut. Jelaskan mengapa?

$$f(n) = \begin{cases} 0, & n = 1 \\ 1 + 2f(n/2), & n \geq 2 \end{cases}$$

- Lakukan substitusi berulang, dimulai dari $f(n) = 1 + 2f(n/2)$ hingga dicapai *base-case* untuk menentukan formula eksplisit dari fungsi $f(n)$.

- Analisislah kasus umum dimana $n \neq 2^k$ (dengan kata lain, n adalah sebarang integer).
- Jika kalkulasi Anda benar, akan diperoleh:

$$f(n) = \begin{cases} 0, & n = 1 \\ f(\lfloor \frac{n}{2} \rfloor) + f(n - \lfloor \frac{n}{2} \rfloor) + 1, & n \geq 2 \end{cases}$$

- Kita dapat membuktikan dengan **teknik induksi** bahwa formula tersebut dapat disederhanakan menjadi:

$$f(n) = n - 1$$

Coba buktikan!

2. (Recursive powering)

Perhatikan algoritma berikut.

Algorithm 5 Recursive powering

```

1: procedure POWER3( $X, n$ )
2:   if  $n = 1$  then
3:     return  $X$ 
4:   end if
5:    $T = \text{POWER3}(X, \lfloor \frac{n}{2} \rfloor)$ 
6:    $T = T * T$ 
7:   if  $n \text{ MOD } 2 = 1$  then
8:      $T = T * X$ 
9:   return  $T$ 
10: end if
11: end procedure

```

$$\triangleright T = T^{\lfloor \frac{n}{2} \rfloor} * T^{\lfloor \frac{n}{2} \rfloor}$$

Kita akan menghitung kompleksitas waktu algoritma tersebut dengan metode yang serupa seperti pada soal sebelumnya.

- Misalkan $f(n)$ adalah fungsi kompleksitas waktunya untuk ukuran input n . Rumuskan formula rekursif dari $f(n)$ dengan memperhitungkan hal-hal berikut ini.
 1. Berapakah banyaknya prosedur perkalian yang dilakukan pada algoritma tersebut?
 2. Tentukan nilai $f(0)$.
 3. Tentukan formula rekursif dari $f(n)$ untuk nilai $n \geq 2$, dengan n ganjil.
 4. Tentukan formula rekursif dari $f(n)$ untuk nilai $n \geq 2$, dengan n genap.
- Perhatikan bahwa ketika nilai n ganjil dan ketika nilai n genap, formula dari $f(n)$ tidak berbeda secara signifikan. Oleh sebab itu, kedua kasus ini dapat digabungkan dengan menggunakan fungsi aproksimasi. Tentukan fungsi aproksimasi yang sesuai.
- Gunakan teknik induksi untuk menghitung formula eksplisit dari $f(n)$.
- Apakah kompleksitas waktu yang Anda dapatkan memiliki perbedaan yang signifikan dengan kompleksitas algoritma POWERING dengan strategi brute-force?