EXERCISE 5: KEBENARAN DAN KOMPLEKSITAS ALGORITMA REKURSIF

untuk dikerjakan sebagai latihan di kelas pada perkuliahan pertemuan ke-5

Petunjuk: Buatlah kelompok beranggotakan 3 orang dan diskusikan permasalahan berikut. Diskusikan sebelum pelaksanaan pertemuan sehingga Anda memiliki pemahaman saat perkuliahan di kelas.

Bagian 1: Kebenaran algoritma rekursif

1. (Menghitung faktorial)

Algorithm 1 Factorial of a number

```
procedure FACTORIAL(n)
    if n = 1 then
        return 1
    else
        temp = FACTORIAL(n − 1)
        return n ∗ temp
    end if
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 $n > 1$.
- Buktikan kebenaran hipotesis Anda. (*Hint: Hipotesis berlaku untuk nilai $n = k$ untuk suatu nilai $k > 1$. 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?

a Mencari nilai maksimum pada array

Algorithm 2 Finding maximum of an array

```
procedure MAX(A[0..n − 1], int n)
    if n = 1 then return A[0]
    else
        T = MAX(A, n − 1)
        if T < A[n − 1] then
            return A[n − 1]
        else
            return T
        end if
    end if
end procedure
```

b Menghitung jumlah elemen pada array

Algorithm 3 Sum of an array

procedure $\text{SUM}(A[0..n-1]$, int $n)$
  if $n = 1$ **then return** $A[0]$
  else
    $S = \text{SUM}(A, n-1)$
    $S = S + A[n-1]$
    if $T < A[n-1]$ **then**
      return S
    end if
  end if
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

procedure $\text{FINDMAX}(A[i..j], n)$ ▷ *i, j are respectively the index of start, end of A*
  if $n = 1$ **then return** $A[0]$
  end if
  $m = \left\lfloor \frac{i+j}{2} \right\rfloor$
  $T_1 = \text{FINDMAX}(A[i..m], \left\lfloor \frac{n}{2} \right\rfloor)$ ▷ *Recursive call the left sub-array*
  $T_2 = \text{FINDMAX}(A\left[(m+1)..j\right], n - \left\lfloor \frac{n}{2} \right\rfloor)$ ▷ *Rec. call right sub-array*
  if $T_1 \geq T_2$ **then return** T_1 ▷ *Compare the two max elements*
  else return T_2
  end if
end procedure

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

```
procedure POWER3(X, n)
    if n = 1 then
        return X
    end if
    T = POWER3(X, ⌊n/2⌋)
    T = T * T                          ▷ T = T^⌊n/2⌋ * T^⌈n/2⌉
    if n MOD 2 = 1 then
        T = T * X
        return T
    end if
end procedure
```

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?