Solution 1: Algorithm Introduction & Divide and Conquer

1 Problem

Let $F(x) = \sum_{k=0\cdots n} a_k x^k$ be a polynomial equation, where $a_0 \cdots a_n$ are stored in an array $A[0 \cdots n]$. Given an input integer x, we can find the corresponding output value $F(x)$. For example, the coefficients of the polynomial $F(x) = 1 + x + 2x^2$ can be stored in an array $< 1, 1, 2 >$. When $x = 2$, $F(x)$ equals to 11.

The basic algorithm (illustrated in Algorithm 1) is designed for solving this problem. The idea is that, for every k (line 2), compute $A[k] \cdot x^k$ (line 3-5) first and then sum up the results (line 7).

Algorithm 1: Basic Algorithm

```
  Input: coefficient array A[0···n] representing a_0···a_n
  Input: x
  Output: corresponding value F(x)
1 sum = 0
2 for k=0 to n do
3     temp = A[k]
4     for i = 1 to k do
5         temp = temp · x
6     sum = sum + temp
7 return sum
```

2 Time Complexity

Analyze the time complexity of Algorithm 1.

Solution:

Table 1 illustrates the time complexity of every line. Sum up the cost of every line gives $O(n^2)$.

Table 1: Time Complexity of Each Line

Line	**Time Complexity**
1	$O(1)$
2	$n+1 = O(n)$
3	same as Line 2
4	$\sum_{k=0..n} k = \frac{n(n+1)}{2} = O(n^2)$
5	same as Line 4
6	same as Line 2
7	$O(1)$

3 Incremental Algorithm

- Design an incremental algorithm for solving the problem.
  hint: $F(x) = \sum_{k=0\cdots n} a_k x^k$ can be reorganized as $F(x) = a_0 + x(a_1 + x(a_2 + \cdots + x(a_{n-1} + x a_n) \cdots))$

- Consider the polynomial equation $F(x) = 4 + 2x + 3x^2 + x^3$. Show the running steps of your algorithm when $x = 2$.
- Analyze the time complexity of your algorithm.

Solution:

- Refer to Algorithm 2.
- Refer to Table 2.
- Time complexity is $O(n)$.

Algorithm 2: Incremental Algorithm

Input: coefficient array $A[0 \cdots n]$ representing $a_0 \cdots a_n$
Input: x
Output: corresponding value $F(x)$

$sum = 0$
for $k = n$ *downto 0* **do**
  $sum = x \cdot sum + A[k]$
return sum

Table 2: Running Steps

k	sum
3	$0 \cdot 2 + A[3] = 1$
2	$1 \cdot 2 + A[2] = 5$
1	$5 \cdot 2 + A[1] = 12$
0	$12 \cdot 2 + A[0] = 28$

Answer to question 4

BinarySearch(A, n_L, n_R, k)

1. **if** $n_L > n_R$
2. return (-1) //not found
3. **else**
4. $mid = (n_L + n_R)/2$
5. if ($k == A[mid]$)
6. return (mid)
7. **else if** ($k < A[mid]$)
8. return *BinarySearch* (A, n_L, mid-1, k)
9. **else**
10. return *BinarySearch* (A, mid+1, n_R, k)

- Let T(n) be the running time
  - **Line 1 and 2**
    Take O(1) time
  - **Line 3 and 4**
    Take O(1) time
  - **Line 5 and 6**
    Take O(1) time
  - **Line 7 to 10**
    Take T(n/2) time

Which is correct?

$T(n) = 2T(n/2) + O(1)$ ✖ $T(n) = T(n/2) + O(1)$ ✔

Answer to question 5

- Solve the recurrence $T(n) = 5T(n/3) + n^2$

  ---- using the **master method**

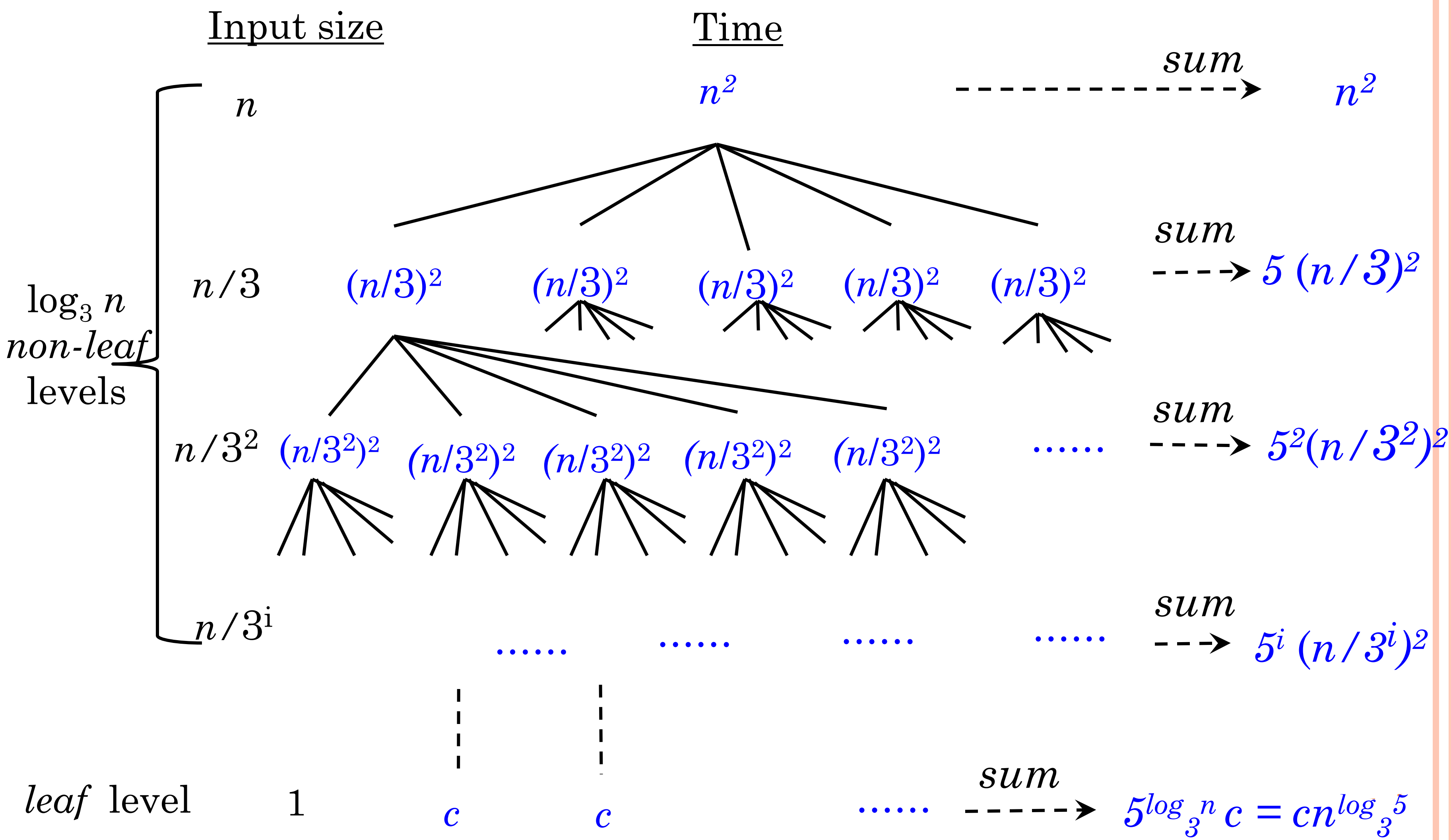
  - $a = 5, \quad b = 3, \longrightarrow n^{\log_b a} = n^{\log_3 5} = n^{1.46}$
  - $f(n) = n^2$
  - Since $f(n) / n^{\log_b a} = \Omega(n^{0.54})$, e.g., use $\varepsilon = 0.54 > 0$,
  - And we have that

    $a\, f(n/b) / f(n) = [5\ (n/3)^2\ ] / n^2 = 5/9 < 1$
  - So we get:

$$\color{red}{T(n) = O(n^2)}$$

---- using the **recursion tree**

Recursion Tree for: $T(n) = 5\,T(n/3) + n^2$



Recursion Tree for: $T(n) = 5\,T(n/3) + n^2$

- Time at **leaf** level: $cn^{\log_3 5} = cn^{1.46}$
- Number of **non-leaf** levels L: $\log_3 n$
- Consider the level i (level 0 is the top level)
  - Input size of a subproblem: $n/3^i$
  - Number of subproblems: 5^i
  - Combine time of a subproblem: $(n/3^i)^2$
  - Total time at this level: $5^i\,(n/3^i)^2$
- Total time $T(n)$:

$$\Sigma_{i=0..L-1}\, 5^i\,(n/3^i)^2 + cn^{1.46}$$

$$= \Sigma_{i=0..\,L-1}\,(5/9)^i\, n^2 + cn^{1.46}$$

$$= O(n^2)$$