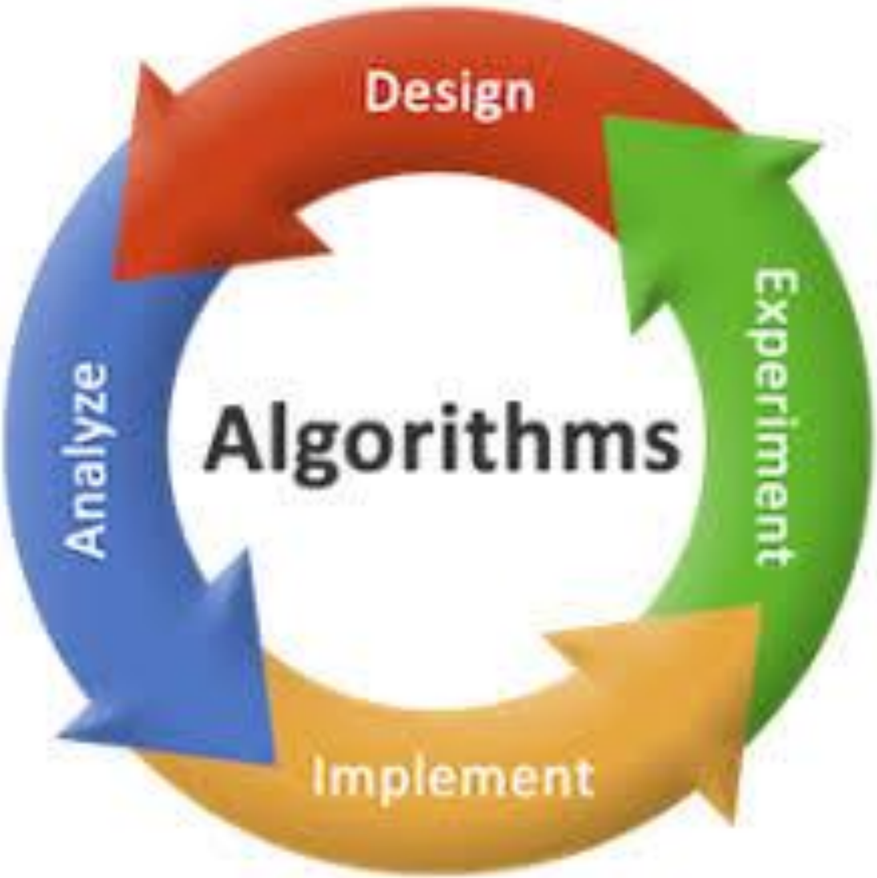

Design & Analysis of Algorithms

Lecture 8

Mathematical Analysis of Recursive Algorithms-I

Mathematical Analysis of Recursive Algorithms

- We systematically apply the general framework (discussed earlier) to analyzing the time efficiency of **recursive** algorithms.

Mathematical Analysis of Recursive Algorithms

General Plan for Analyzing the Time Efficiency

- **1.** Decide on a **parameter** (or parameters) indicating an input's size.
- **2.** Identify the algorithm's **basic operation**.
- **3.** Check whether the number of times the basic operation is executed can vary on different inputs of the same size; if it can, the **worst-case**, **average-case**, and **best-case** efficiencies must be investigated separately.

Mathematical Analysis of Recursive Algorithms

General Plan for Analyzing the Time Efficiency

- **4.** Set up a **recurrence relation**, with an appropriate **initial condition**, for the number of times the basic operation is executed.
- **5. Solve** the recurrence or, at least, ascertain the **order of growth** of its solution.

Mathematical Analysis of Recursive Algorithms

Example 1

- Compute the factorial function $F(n) = n!$ for an arbitrary nonnegative integer n.

Since $n! = 1 \,.\,.\,.\,.\,.\,. (n - 1) \,.\, n = (n - 1)! \,.\, n$ for $n \geq 1$ and $0! = 1$ by definition, we can compute $\mathbf{F(n) = F(n - 1) \,.\, n}$ with the following recursive algorithm.

Mathematical Analysis of Recursive Algorithms

ALGORITHM $F(n)$

//Computes $n!$ recursively
//Input: A nonnegative integer n
//Output: The value of $n!$
if $n = 0$ **return** 1
else return $F(n-1) * n$

- For simplicity, we consider ***n*** as an indicator of this algorithm's input size.

Mathematical Analysis of Recursive Algorithms

- The **basic operation** of the algorithm is **multiplication**, whose number of executions we denote **M(n)**.
- Since the function **F(n)** is computed according to the formula **F(n) = F(n − 1) . n** for **n > 0**, the number of multiplications **M(n)** needed to compute it must satisfy the equality

$$M(n) = \underset{\substack{\text{to compute} \\ F(n-1)}}{M(n-1)} + \underset{\substack{\text{to multiply} \\ F(n-1) \text{ by } n}}{1} \qquad \text{for } n > 0$$

Mathematical Analysis of Recursive Algorithms

- This equation defines **M(n) not explicitly**, i.e., as a function of **n**, but **implicitly** as a function of its value at another point, namely **n−1**.
- Such equations are called **recurrence relations** or **recurrences**.
- Our **goal** now is to **solve** the **recurrence relation** $M(n) = M(n-1) + 1$, i.e., to find an explicit formula for **M(n)** in terms of **n** only.

Mathematical Analysis of Recursive Algorithms

- To determine a solution uniquely, we need an **initial condition** that tells us the value with which the sequence starts.
- We can obtain this value by inspecting the condition that makes the algorithm stop its recursive calls:

if n == 0 return 1.

Mathematical Analysis of Recursive Algorithms

- **First**, since the calls stop when n = 0, the smallest value of n for which this algorithm is executed and hence M(n) defined is 0.
- **Second**, by inspecting the pseudocode's exiting line, we can see that when n = 0, the algorithm performs no multiplications.
- Therefore, the initial condition is:

$$M(0) = 0.$$

the calls stop when $n = 0$ no multiplications when $n = 0$

Mathematical Analysis of Recursive Algorithms

- Thus, the recurrence relation and initial condition for the algorithm's number of multiplications M(n):

$$M(n) = \begin{cases} M(n-1) + 1 \text{ for } n > 0, \\ 0 \text{ for } n=0. \end{cases}$$

- We use the method of **backward substitutions** to solve this recurrence relation.

Mathematical Analysis of Recursive Algorithms

- **Solution**

$$M(n) = M(n-1) + 1 \qquad \text{substitute } M(n-1) = M(n-2) + 1$$
$$= [M(n-2) + 1] + 1 = M(n-2) + 2 \qquad \text{substitute } M(n-2) = M(n-3) + 1$$
$$= [M(n-3) + 1] + 2 = M(n-3) + 3.$$

- Of the general form M(n) = M(n − i) + i

$$M(n) = M(n-1) + 1 = \ldots$$
$$= M(n-i) + i = \ldots$$
$$= M(n-n) + n = \boldsymbol{n}.$$

References

- **Chapter 2**: Anany Levitin, “Introduction to the Design and Analysis of Algorithms”, Pearson Education, Third Edition, 2017

- **Chapter 2**: Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein, “Introduction to Algorithms”, MIT Press/PHI Learning Private Limited, Third Edition, 2012.

Homework

Analyze following recursive algorithm:

The determinant of an $n \times n$ matrix, $A = \begin{bmatrix} a_{0\,0} & \cdots & a_{0\,n-1} \\ a_{1\,0} & \cdots & a_{1\,n-1} \\ \vdots & & \vdots \\ a_{n-1\,0} & \cdots & a_{n-1\,n-1} \end{bmatrix}$,

denoted by **det A**, can be defined as a_{00} for n = 1 and, for n>1, by the recursive formula, $\det A = \sum_{j=0}^{n-1} s_j a_{0\,j} \det A_j$,

where s_j is +1 if j is even and −1 if j is odd, a_{0j} is the element in row 0 and column j, and A_j is the $(n-1) \times (n-1)$ matrix obtained from matrix A by deleting its row 0 and column j.