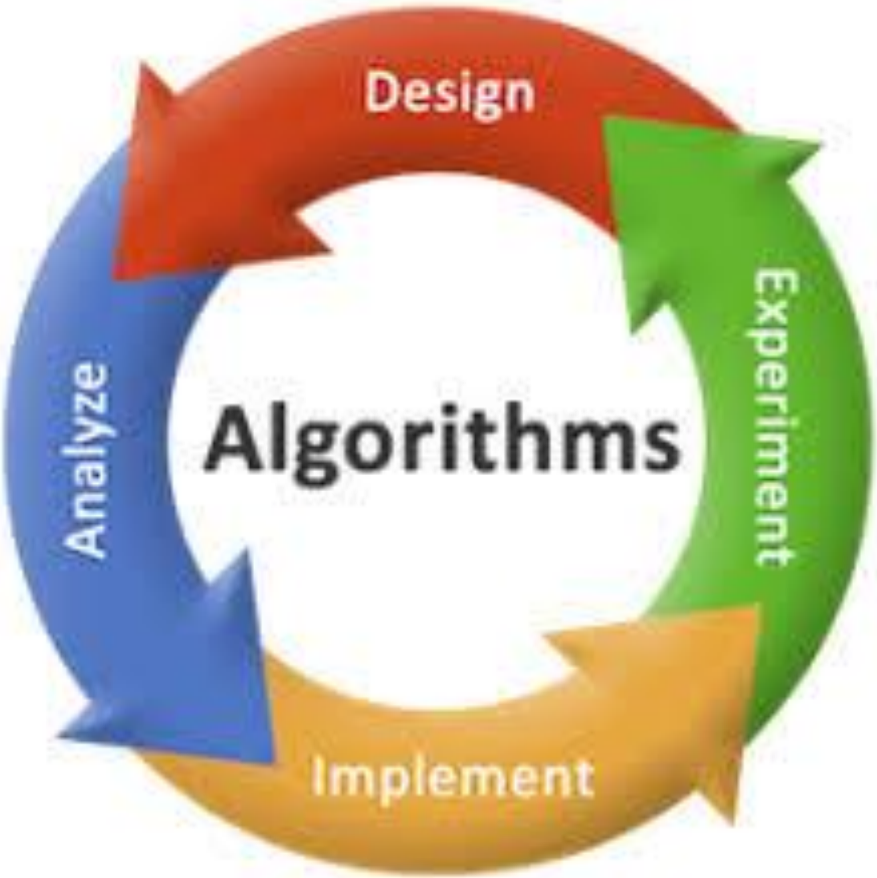

Design & Analysis of Algorithms

Lecture 7

Mathematical Analysis of Non-recursive Algorithms

Mathematical Analysis of Non-recursive Algorithms

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

Mathematical Analysis of Non-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**. (As a rule, it is located in the **innermost** loop.)
- **3.** Check whether the **number of times** the **basic operation** is executed depends only on the **size** of an **input**. If it also depends on some additional property, the worst-case, average-case, and, if necessary, best-case efficiencies have to be investigated separately.

Mathematical Analysis of Non-recursive Algorithms

General Plan for Analyzing the Time Efficiency

- **4.** Set up a **sum** expressing the number of times the algorithm's basic operation is executed.
- **5.** Using standard formulas and rules of sum manipulation, either find a closed-form formula for the count or, at the very least, establish its **order of growth**.

Mathematical Analysis of Non-recursive Algorithms

Example 1

- Consider the problem of finding the value of the largest element in a list of n numbers.

- For simplicity, we assume that the list is implemented as an **array**.

Mathematical Analysis of Non-recursive Algorithms

ALGORITHM $MaxElement(A[0..n-1])$

//Determines the value of the largest element in a given array
//Input: An array $A[0..n-1]$ of real numbers
//Output: The value of the largest element in A
$maxval \leftarrow A[0]$
for $i \leftarrow 1$ **to** $n-1$ **do**
    if $A[i] > maxval$
        $maxval \leftarrow A[i]$
return $maxval$

- input's size here is the number of elements in the array, i.e., *n.*

Mathematical Analysis of Non-recursive Algorithms

- Algorithm's for loop-
  - There are two operations in the loop's body: the comparison A[i]> maxval and
  - the assignment maxval←A[i].

- Which of these two operations should we consider **basic**?

Mathematical Analysis of Non-recursive Algorithms

- Since the comparison is executed on each repetition of the loop and the assignment is not, we should consider the **comparison** to be the algorithm's basic operation.

- Note that the number of comparisons will be the same for all arrays of size n; therefore, in terms of this metric, there is **no need** to distinguish among the **worst**, **average**, and **best** cases here.

Mathematical Analysis of Non-recursive Algorithms

- Let us denote C(n) the number of times this comparison is executed.
- The algorithm makes one comparison on each execution of the loop, which is repeated for each value of the loop's variable i within the bounds 1 and n − 1, inclusive.
- Therefore, we get the following sum for C(n):

$$C(n) = \sum_{i=1}^{n-1} 1$$

$$C(n) = \sum_{i=1}^{n-1} 1 = n - 1 \in \Theta(n)$$

Mathematical Analysis of Non-recursive Algorithms

EXAMPLE 2

- Consider the element uniqueness problem: check whether all the elements in a given array of n elements are distinct.

- This problem can be solved by the following straightforward algorithm.

Mathematical Analysis of Non-recursive Algorithms

ALGORITHM *UniqueElements*$(A[0..n-1])$

//Determines whether all the elements in a given array are distinct
//Input: An array $A[0..n-1]$
//Output: Returns "true" if all the elements in A are distinct
//   and "false" otherwise
for $i \leftarrow 0$ **to** $n-2$ **do**
    for $j \leftarrow i+1$ **to** $n-1$ **do**
        if $A[i] = A[j]$ **return false**
return true

- input's size here is again n, the number of elements in the array.

Mathematical Analysis of Non-recursive Algorithms

- Since the innermost loop contains a single operation (the **comparison** of two elements), we should consider it as the algorithm's **basic** operation.
- **Note**, however, that the number of comparisons depends not only on **n** but also on whether there are equal elements in the array and, if there are, which **array positions** they occupy.
- Let us investigate the **worst case** scenario.

Mathematical Analysis of Non-recursive Algorithms

$$C_{worst}(n) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1 = \sum_{i=0}^{n-2} [(n-1) - (i+1) + 1] = \sum_{i=0}^{n-2} (n-1-i)$$

$$= \sum_{i=0}^{n-2} (n-1) - \sum_{i=0}^{n-2} i = (n-1) \sum_{i=0}^{n-2} 1 - \frac{(n-2)(n-1)}{2}$$

$$= (n-1)^2 - \frac{(n-2)(n-1)}{2} = \frac{(n-1)n}{2} \approx \frac{1}{2} n^2 \in \Theta(n^2).$$

Mathematical Analysis of Non-recursive Algorithms

Some formula to consider

$$\sum_{i=l}^{u} ca_i = c \sum_{i=l}^{u} a_i,$$

$$\sum_{i=l}^{u} (a_i \pm b_i) = \sum_{i=l}^{u} a_i \pm \sum_{i=l}^{u} b_i,$$

$$\sum_{i=l}^{u} 1 = u - l + 1 \quad \text{where } l \le u \text{ are some lower and upper integer limits,}$$

$$\sum_{i=0}^{n} i = \sum_{i=1}^{n} i = 1 + 2 + \cdots + n = \frac{n(n+1)}{2} \approx \frac{1}{2}n^2 \in \Theta(n^2).$$

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 the following algorithms:

- Matrix multiplication of dimensions n x n.
- Finding the number of binary digits in the binary representation of a positive decimal integer.