# Design and Analysis of Algorithms

**Vandana M L**
Department of Computer Science & Engineering

# DESIGN AND ANALYSIS OF ALGORITHMS

## Mathematical Analysis of Non-recursive Algorithms

Slides courtesy of **Anany Levitin**

**Vandana M L**

Department of Computer Science & Engineering

## Time Efficiency of Non-recursive Algorithms

Steps in mathematical analysis of non-recursive algorithms:

- Decide on parameter n indicating input size
- Identify algorithm's basic operation
- Check whether the number of times the basic operation is executed depends only on the input size n. If it also depends on the type of input, investigate worst, average, and best case efficiency separately.
- Set up summation for C(n) reflecting the number of times the algorithm's basic operation is executed.
- Simplify summation using standard formulas

## Useful Summation Formulas and Rules

$\Sigma_{l \le i \le u} 1 = 1+1+\ldots+1 = u - l + 1$

$\Sigma_{1 \le i \le n}\, i = 1+2+\ldots+n = n(n+1)/2$

$\Sigma_{1 \le i \le n}\, i^2 = 1^2+2^2+\ldots+n^2 = n(n+1)(2n+1)/6$

$\Sigma_{0 \le i \le n}\, a^i = 1 + a + \ldots + a^n = (a^{n+1} - 1)/(a - 1)$ for any $a \ne 1$

$\Sigma(a_i \pm b_i) = \Sigma a_i \pm \Sigma b_i \qquad \Sigma c a_i = c \Sigma a_i$

$\Sigma_{l \le i \le u} a_i = \Sigma_{l \le i \le m} a_i + \Sigma_{m+1 \le i \le u} a_i$

$$\sum_{i=l}^{u} 1 = (u - l + 1)$$

## Example 1: Finding Max Element in a list

```
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 ← A[0]
for i ← 1 to n-1 do
        if A[i] > maxval
                maxval ← A[i]
return maxval
```

- The basic operation- comparison
- Number of comparisons is the same for all arrays of size n.
- Number of comparisons

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

## Example 2: Element Uniqueness Problem

**Algorithm** *UniqueElements (A[0..n-1])*
//Checks 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 ← 0 to n - 2 do
        for j ← i + 1 to n – 1 do
                if A[i] = A[j] return false
return true
```

Best-case:

If the two first elements of the array are the same
No of comparisons in Best case = 1 comparison

Worst-case:

- Arrays with no equal elements
- Arrays in which only the last two elements are the pair of equal elements

## Example 2: Element Uniqueness Problem

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

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

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

Best-case: 1 comparison

Worst-case: $n^2/2$ comparisons

$T(n)_{\text{worst case}} = O(n^2)$

## Example 3:Matrix Multiplication

**Algorithm** *MatrixMultiplication(A[0..n-1, 0..n-1], B[0..n-1, 0..n-1] )*

```
//Multiplies two square matrices of order n by the definition-based algorithm
//Input: two n-by-n matrices A and B
//Output: Matrix C = AB
for i ← 0 to n - 1 do
   for j ← 0 to n – 1 do
            C[i, j] ← 0.0
             for k ← 0 to n – 1 do
                         C[i, j] ← C[i, j] + A[i, k] * B[k, j]
return C
```

$M(n) \in \Theta(n^3)$

# THANK YOU

**Vandana M L**
Department of Computer Science & Engineering
**vandanamd@pes.edu**