UNIT I INTRODUCTION:

Introduction to Algorithms – Fundamentals of Algorithmic Problem Solving – Fundamentals of the Analysis of Algorithmic Efficiency – Analysis Framework – Asymptotic Notations and Basic Efficiency Classes – Mathematical Analysis of Recursive Algorithms – Mathematical Analysis of Non-recursive Algorithms

Mathematical Analysis of Non-recursive Algorithms

Time efficiency of non-recursive algorithms

General Plan for Analysis

- **Decide on parameter *n* indicating *input size***
- **Identify algorithm's *basic operation***
- **Determine *worst*, *average*, and *best* cases for input of size *n***
- **Set up a sum for the number of times the basic operation is executed**
- **Simplify the sum using standard formulas and rules**

Useful summation formulas and rules

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

In particular, $\Sigma_{1 \le i \le n} 1 = n - 1 + 1 = n \in \Theta(n)$

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

$\Sigma_{1 \le i \le n}\, i^2 = 1^2+2^2+\ldots+n^2 = n(n+1)(2n+1)/6 \approx n^3/3 \in \Theta(n^3)$

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

In particular, $\Sigma_{0 \le i \le n}\, 2^i = 2^0 + 2^1 + \ldots + 2^n = 2^{n+1} - 1 \in \Theta(2^n)$

$\Sigma(a_i \pm b_i) = \Sigma a_i \pm \Sigma b_i$ $\quad \Sigma c a_i = c \Sigma a_i$ $\quad \Sigma_{l \le i \le u} a_i = \Sigma_{l \le i \le m} a_i + \Sigma_{m+1 \le i \le u} a_i$

Example 1: Maximum element

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 $maxval$

$T(n) = \sum_{1 \le i \le n-1} 1 = n-1 = \Theta(n)$ comparisons

Example 2: Element uniqueness problem

```
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 ← 0 to n − 2 do
        for j ← i + 1 to n − 1 do
            if A[i] = A[j] return false
    return true
```

T(n) = $\sum_{0 \le i \le n-2} (\sum_{i+1 \le j \le n-1} 1)$

= $\sum_{0 \le i \le n-2}$ n-i-1 = (n-1+1)(n-1)/2

= $\Theta(n^2)$ comparisons

Example 3: Matrix multiplication

```
ALGORITHM  MatrixMultiplication(A[0..n − 1, 0..n − 1], B[0..n − 1, 0..n − 1])
    //Multiplies two n-by-n matrices 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
```

$T(n) = \sum_{0 \le i \le n-1} \sum_{0 \le i \le n-1} n$

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

$= \Theta(n^3)$ multiplications

Example 4: Gaussian elimination

Algorithm *GaussianElimination*(A[0..n-1,0..n])
//Implements Gaussian elimination of an n-by-(n+1) matrix A

for $i \leftarrow 0$ to $n - 2$ do
  for $j \leftarrow i + 1$ to $n - 1$ do
    for $k \leftarrow i$ to n do
      $A[j,k] \leftarrow A[j,k] - A[i,k] * A[j,i] / A[i,i]$

for $i \leftarrow 0$ **to** $n - 2$ **do**
  for $j \leftarrow i + 1$ **to** $n - 1$ **do**
    $B \leftarrow 0$
    for $k \leftarrow i$ **to** n **do**
      $B \leftarrow A[i,k] * A[j,i]$
    $A[j,k] \leftarrow A[j,k] - B / A[i,i]$

Find the efficiency class and a constant factor improvement.

Example 5: Counting binary digits

ALGORITHM $Binary(n)$

//Input: A positive decimal integer n
//Output: The number of binary digits in n's binary representation
$count \leftarrow 1$
while $n > 1$ **do**
  $count \leftarrow count + 1$
  $n \leftarrow \lfloor n/2 \rfloor$
return $count$

The halving game: Find integer i such that $n/2^i \leq 1$.

Answer:* $i \leq \log n$.** ***So, $T(n) = \Theta(\log n)$ divisions.

Another solution: Using recurrence relations.

Thank You