Algorithm/Running Time Analysis

Running Time

- Why do we need to analyze the running time of a program?
- Option 1: Run the program and time it
  - Why is this option bad?
  - What can we do about it?

Pseudo-Code

- Used to specify algorithms
- Part English, part code

```
Algorithm (arrayMax(A, n))
curMax = A[0]
for i=1 i<n i++
  if curMax < A[i]
    curMax = A[i]
return curMax
```

Math Review

- Summation $\sum$
- Sum of n consecutive digits = $n(n+1)/2$

Counting Operations

```
Algorithm (arrayMax(A, n))
curMax = A[0] //1
for i=1 i<n i++ //n
  //1 or 2
  if curMax < A[i]
    curMax = A[i]
return curMax //1
```

- Best case – n+2
- Worst case – 2n + 2
- Average case – hard to analyze

Asymptotic Notation

- $2n + 2$
- n=5 -> 12
- n=100 -> 202
- n=1,000,000 -> 2,000,002

- Running time grows proportionally to n
- What happens as n gets large?
Big-Oh
• \( f(n) \) is \( O(g(n)) \) if there is a real constant \( c > 0 \) and an integer constant \( n_0 \geq 1 \) such that \( f(n) \leq cg(n) \) for every integer \( n \geq n_0 \)
• \( 2n+2 \) is \( O(n) \) \( n_0 \geq 1 \) \( c = 3 \)

Examples
• \( 87n^4 + 7n \)
• \( 3n \log n + 12 \log n \)
• \( 4n^4 + 7n^3 \log n \)

Terminology
• Logarithmic – \( O(\log n) \)
• Linear – \( O(n) \)
• Linearithmic – \( O(n \log n) \)
• Quadratic – \( O(n^2) \)
• Polynomial – \( O(n^k) \) \( k \geq 1 \)
• Exponential – \( O(a^n) \) \( a > 1 \)

Example
• What is the big-oh running time of this algorithm?

Algorithm: 
Input: \( A, n \)
    curMax = \( A[0][0] \)
    for i=0 i<n i++
        for j=0 j<n j++
            if curMax < \( A[i][j] \)
                curMax = \( A[i][j] \)
    return curMax

Another Example
• Determine how many elements of array 1 match elements of array 2

Algorithm?
Another Example

Input: A, B, n
for i = 0 to n-1
  for j = 0 to n-1
    if A[i] == A[j]
      matches++
      break

• What is the running time of the algorithm?