COSC 2123/1285 Algorithms and Analysis
Tutorial 2
Fundamentals of Algorithms Analysis

Objective

Students who complete this tutorial should:

- Understand the fundamentals of algorithm analysis.
- Have a sound understanding of the analysis framework used to evaluate the efficiency of algorithms.
- Be familiar with asymptotic complexity.
- Be able to evaluate recursive and non-recursive algorithms.

Questions

2.1.3 Consider a variation of sequential search that scans a list to return the number of occurrences of a given search key in the list. Will its efficiency differ from the efficiency of classic sequential search?

2.1.9 Indicate whether the first function of each of the following pairs has a smaller, same or larger order of growth (to within a constant multiple) than the second function.

a. $100n^2$ and $0.01n^3$.

b. $\log_2(n)$ and $\ln(n)$.

c. $(n-1)!$ and $n!$.

2.2.2 Use the informal definitions of O, Θ and Ω to determine whether the following assertions are true or false.

a. $\frac{n(n+1)}{2} \in O(n^3)$.

b. $\frac{n(n+1)}{2} \in O(n^2)$.

c. $\frac{n(n+1)}{2} \in \Theta(n^3)$.

d. $\frac{n(n+1)}{2} \in \Omega(n)$.

2.2.5 Order the following functions according to their order of growth (from the lowest to the highest):

$$(n-2)!,\ 5\log(n+100)^{10},\ 2^{2n},\ 0.0001n^4 + 3n^3 + 1,\ \ln^2(n),\ \sqrt[3]{n},\ 3^n$$

2.3.1 Compute the following sum:

$$\sum_{i=3}^{n+1} 1$$

2.3.4 Consider ALGORITHM 1.

Algorithm 1 Mystery(n)

```
// Input: a non-negative integer n
S = 0
for i = 1 to n do do
    S = S + i * i
end for
return S
```

a. What does this algorithm compute?

b. What is its basic operation?

c. How many times is the basic operation executed?

d. What is the efficiency class of this algorithm?

e. Suggest an improvement or a better algorithm altogether and indicate its efficiency class. If you cannot do it, try to prove that in fact it cannot be done.

2.4.1 Solve the following recurrence relations:

a. $x(n) = x(n-1) + 1$, for $n > 0$, $x(0) = 0$

2.4.4 Consider ALGORITHM 2.

Algorithm 2 $Q(n)$

```
// Input: a positive integer n
if n == 1 then
    return 1
else
    return Q(n - 1) + 2 * n - 1
end if
```

a. Set up a recurrence relation for this function's return values. Make a guess on what the function computes.

b. Set up a recurrence relation for the number of multiplications made by the algorithm.

