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 big O notation.
- 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?

Answer: The algorithm will always make n key comparisons on every input of size n, whereas this number vary between n and 1 for the classic version of 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 $\log_e(n)$.

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

Answer:

1. $100n^2$ (quadratic) has a lower order of growth than $0.01n^3$ (cubic).
2. Since changing a logarithm's base can be done by the formula

   $$\log_a(n) = \log_a(b) \log_b(n)$$

   all logarithmic functions have the same order of growth to within a constant multiple.
3. $(n-1)!$ has a lower order of growth than $n! = (n-1)!n$.

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

Answer: $\frac{n(n+1)}{2} \approx \frac{n^2}{2}$ is quadratic. Therefore

1. $\frac{n(n+1)}{2} \in O(n^3)$ is true.
2. $\frac{n(n+1)}{2} \in O(n^2)$ is true.
3. $\frac{n(n+1)}{2} \in \Theta(n^3)$ is false.
4. $\frac{n(n+1)}{2} \in \Omega(n)$ is true.

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

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

Answer: First, simplify the functions given. Then use the list of functions in Table 2.2(book) to simplify each of the functions given. Prove their final placement by computing appropriate limits (get rid of constants etc).

1. $(n-2)! \in \Theta((n-2)!)$
2. $5\log(n+100)^{10} = 50\log(n+100) \in \Theta(\log(n))$
3. $2^{2n} \in \Theta((2^2)^n)$
4. $0.0001n^4+3n^3+1 \in \Theta(n^4)$
5. $\ln^2(n) \in \Theta(\log^2(n))$
6. $\sqrt[3]{n} \in \Theta(n^{\frac{1}{3}})$
7. $3^n \in \Theta(3^n)$

The list of these functions ordered in increasing order of growth looks as follows:

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

2.3.1 Compute the following sum (extension):

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

Answer: Use the common summation formulas and rules listed in Appendix A (book). You may need to perform some simple algebraic operations before

applying them.

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

2.3.4 Consider ALGORITHM 1.

Algorithm 1 Mystery(n)

```
// Input: a non-negative integer n
S = 0
for i = 1 to n do
    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.

Answer:

1. Tracing the algorithm to get its output for a few small values of n should help if you need it. Computes $S(n) = \sum_{i=1}^{n} i^2$.
2. Multiplication (or, if multiplication and addition are assumed to take the same amount of time, either of the two).
3. $C(n) = \sum_{i=1}^{n} 1 = n$.
4. $C(n) = n \in \Theta(n)$.
5. Use the formula $S(n) = \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$ to compute the sum in $\Theta(1)$ (constant) time.

2.4.1 Solve the following recurrence relations:

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

Answer: Use backward substition.

$$\begin{aligned} x(n) &= x(n-1) + 1 \\ &= [x(n-2) + 1] + 1 = x(n-2) + 2 \\ &= [x(n-3) + 1] + 2 = x(n-3) + 3 \\ &= \dots \\ &= x(n-i) + i \\ &= \dots \\ &= x(0) + n = 0 + n = n \end{aligned}$$

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.

Answer:

1. Note that you are asked here about a recurrence of the function's values, not about a recurrence for the number of times its operation is executed. Just follow the pseudocode to set it up. We use backward substitution to solve this (note, it will be great if you can solve recurrences of this difficulty yourself, but you are not expected to. We are more interested in you understanding the procedure and how it is used for theoretical analysis).

$$\begin{aligned} Q(n) &= Q(n-1) + 2 * n - 1 \\ &= [Q(n-2) + 2 * (n-1) - 1] + 2 * n - 1 = Q(n-2) + 4 * n - 4 \\ &= [Q(n-3) + 2 * (n-2) - 1] + 4 * n - 4 = Q(n-3) + 6 * n - 9 \\ &= [Q(n-4) + 2 * (n-3) - 1] + 6 * n - 9 = Q(n-4) + 8 * n - 16 \\ &= \dots \\ &= Q(n-i) + 2 * i * n - i^2 \\ &= \dots \\ &= Q(1) + 2 * (n-1) * n - (n-1)^2 = 1 + 2 * n^2 - 2n - n^2 + 2n - 1 = n^2 \end{aligned}$$

2. This question is very similar to one we have already discussed. $M(n) = M(n-1) + 1$ for $n > 1$, $M(1) = 0$. Solving it by backward substitution yields $M(n) = n - 1$.

