

CIS121 - Fall 2008

Lab 2 – Monday/Tuesday, September 15/16

1

Show that $8n^2 + 10n + 25$ is not $O(n)$.

Solution: To obtain a contradiction suppose there are constants c and n_0 such that

$$8n^2 + 10n + 25 \leq cn \quad \forall n \geq n_0$$

Clearly, c has to be larger than 10. So let us assume this. Then, the above inequality implies

$$8n^2 \leq (c - 10)n - 25 \quad \forall n \geq n_0$$

Now pick $n = k(c - 10)$ where k is a natural number such that $k(c - 10) \geq n_0$. Then the $LHS = 8k^2(c - 10)^2$ and the $RHS = k(c - 10)^2 - 25$. Clearly, the LHS is larger than the RHS - a contradiction.

2

Suppose that $f(x)$, $g(x)$ and $h(x)$ are functions such that $f(x)$ is $O(g(x))$ and $g(x)$ is $O(h(x))$. Show that $f(x)$ is $O(h(x))$.

Solution: There are constants c_1, c_2, n_1 and n_2 such that $f(x) \leq c_1g(x)$ for all $x \geq n_1$ and $g(x) \leq c_2h(x)$ for all $x \geq n_2$. therefore for $n_3 \geq \max(n_1, n_2)$ it follows that $f(x) \leq c_1g(x) \leq c_1c_2h(x)$ for all $x \geq n_3$. This shows that $f(x)$ is $O(h(x))$

3

Prove that the factorial function $n!$ is $O(n^n)$. Remember the factorial function $f(n) = n!$ is defined by

$$n! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot n$$

when n is a positive integer, and $0! = 1$.

Solution: A big-Oh estimate for $n!$ can be obtained by noting that each term in the product does not exceed n . Hence,

$$\begin{aligned} n! &= 1 \cdot 2 \cdot 3 \cdot \dots \cdot n \\ &\leq n \cdot n \cdot n \cdot \dots \cdot n \\ &= n^n \end{aligned}$$

This inequality shows that $n!$ is $O(n^n)$ when $c = 1$ and $n_0 = 1$.

4

True or false:

1. 2^{n+1} is $O(2^n)$.

Solution: True.

$2^{n+1} = 2 \cdot 2^n$ meaning $2^{n+1} \leq c2^n$ for $c = 2$ and $n \geq 1$.

2. 2^{2n} is $O(2^n)$.

Solution: False.

Assume by negation that 2^{2n} is $O(2^n)$ then there exist c and n_0 such that $2^{2n} \leq c2^n$ for all $n > n_0$. However if $2^{2n} \leq c2^n$ then $2^n \leq c$ for all $n > n_0$. In other words $n \leq \log_2(c)$ for all $n > n_0$, impossible as $\log_2(c)$ is a constant.

5

For each of the following, if true prove using the definition of big-Oh bound, if false give a counterexample.

1. $g(n)$ is $O(nf(n))$, where $g(n) = \sum_{i=1}^n f(i) = f(1) + f(2) + \dots + f(n)$, and $f(n)$ positive.

Solution: Let $f(n) = \frac{1}{2^n}$. Then:

$$g(n) = \frac{1}{2} + \frac{1}{2^2} + \dots + \frac{1}{2^n} = \frac{1}{2} \left(\frac{1 - \frac{1}{2^n}}{1 - \frac{1}{2}} \right) = 1 - \frac{1}{2^n} = \frac{2^n - 1}{2^n}$$

Assume $g(n)$ is $O(nf(n))$. Then for all $n \geq n_0$ and a positive constant c the following inequality should hold:

$$\frac{2^n - 1}{2^n} \leq c \frac{n}{2^n}$$

which is equivalent to

$$2^n - 1 \leq cn$$

The latter is not true for large n .

2. For any positive $f(n)$ and $g(n)$, if $\log(f(n))$ is $O(\log(g(n)))$ then $f(n)$ is $O(g(n))$.

Solution: Let $f(n) = n^2$ and $g(n) = n$. Then $\log(f(n)) = 2 \log n$, while $\log(g(n)) = \log n$. In this case we have $\log(f(n))$ is $O(\log(g(n)))$ for all $n \geq 1$ and constant $c = 2$. However, we know n^2 is *not* $O(n)$.

6

Give a big-Oh characterization for the running time of the code fragments below, with a short explanation:

```

a) for(int i = 0; i < n; i++ )
    for( int j = i; j <= n; j++ )
        for( int k = i; k <= j; k++ )
            sum++;

```

Solution:The innermost loop runs in time $O(j - i)$, the middle loop runs in time $O((n - i)^2)$ therefore the whole code fragment runs in time $O(n^3)$. More details (not required for the solution)

$$\sum_{i=0}^{n-1} \sum_{j=i}^{n-1} c(j-i+1) = \sum_{i=0}^{n-1} c \frac{(n-i+1)(n-1+2)}{2} = \sum_{k=1}^n \frac{(k+1)(k+2)}{2} = \frac{c}{2} \left(\sum_{k=1}^n k^2 + 3k + 2 \right) = O(n^3)$$

```

b) static int foo(int[] a) {
    int s=a[0];
    for(int i=2;i<=n;i=i+2)
        s=s+a[i];
    return s;
}

```

Solution: $O(n)$. The code snippet contains a single loop, which executes in each iteration a constant number of operations and which repeats $n/2$ times.

```

c) for( int p = 0; p < n*n; p++ )
    for( int q = 0; q < p; q++ )
        sum++;

```

Solution: $O(n^4)$. The inner loop runs takes $O(p)$ and p runs from 0 to n^2 . Therefore the whole fragment runs in time $O((n^2)^2)$

$$1 + 2 + \dots + n + \dots + n^2 = \frac{n^2(n^2 + 1)}{2}$$