Anissa Cristerna

Lab 3

Comp 271

2/12/2019

3.3

**a.** For the inner **for** statement, when i = 0, j takes on values from 1 to n - 1, and so there are n - 1

iterations of the inner **for** statement when i = 0. How many iterations are there when i = 1? When

i = 2?

n – 2 when i = 1

n – 3 when i = 2

**b.** Determine, as a function of n, the total number of iterations of the inner **for** statement as i takes on

values from 0 to n – 2.

Total # iterations = (n - 1) + (n - 2) + … + 1 =

(n – 1) (n)

\_\_\_\_\_\_\_\_\_

2

**c.** Use Big-O notation to estimate worstTime(*n*). In plain English, estimate worstTime(*n*)—the choices are

constant, logarithmic in *n*, linear in *n*, linear-logarithmic in *n*, quadratic in *n* and exponential in *n*.

O ( n^2)

**3.4** For each of the following functions *f*, where *n* = 0, 1, 2, 3, *. . .* , estimate *f* using Big-O notation and plain

English:

**a.** *f* (*n*) = (2 + *n*) \* (3 + log(*n*))

O (n log n)

**b.** *f* (*n*) = 11 \* log(*n*) + *n*/2 − 3452

O (n)

**c.** *f* (*n*) = 1 + 2 + 3 +· · · + *n*

O (n^2)

**d.** *f* (*n*) = *n* \* (3 + *n*) − 7 \* *n*

O (n^2)

**e.** *f* (*n*) = 7 \* *n* + (*n* − 1) \* log (*n* − 4)

O (n log n)

**f.** *f* (*n*) = log (*n^*2*)*+ *n*

O (n)

**g.** *f (n)* = *(n* + 1*)* ∗ log*(n* + 1*)* − *(n* + 1*)* + 1 */ n*

O (log n)

**h.** *f* (*n*) = n + *n*/2 + *n*/4 + *n*/8 + *n*/16 + ·· ·

O (n)

1. Suppose a method foo2() has a running time-complexity of O(n logn). Consider

the method foo1(..) as below:

Public void foo1() {

For (int i=0;i<n;i++) {

Foo2();

}

}

What is the time-complexity of foo1() in big-Oh notation?

1. n^2 log n