Name: \_\_Daniel Detore\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_9/25/2024\_\_\_\_\_\_

*I pledge my honor that I have abided by the Stevens Honor System.*

Point values are assigned for each question. Points earned: \_\_\_\_ / 100, = \_\_\_\_ %

1. Find a tight upper bound for . Write your answer here: \_\_**O(n4)**\_\_\_ (4 points)

Prove your answer by giving values for the constants and . Choose the smallest integer value possible for . (4 points)

**c = 2, n0 = 4**

1. Find an asymptotically tight bound for . Write your answer here: (4 points)

Prove your answer by giving values for the constants , , and . Choose the tightest integer values possible for and . (6 points)

1. Is Circle your answer: yes / ***no***. (2 points)

If yes, prove your answer by giving values for the constants and . Choose the smallest integer value possible for . If no, derive a contradiction. (4 points)

If , then where ().

We also have

Combining these statements gives us:

Because n can grow to infinity, there is no way to bound it by any constant . .

1. Write the following asymptotic efficiency classes in **increasing** order of magnitude.

, , , , , , , , (2 points each)

**O(1), O(lg n), O(n), O(nlogn), O(n2), O(n2logn), O(n3), O(2n), O(n!), O(nn)**

1. Determine the largest size *n* of a problem that can be solved in time *t*, assuming that the algorithm takes *f(n)* milliseconds. Write your answer for *n* as an integer. (2 points each)
2. *f(n)* = , *t* = 1 second n = 1000
3. *f(n)* = , *t* = 1 hour n = 204094
4. *f(n)* = , *t* = 1 hour n = 1897
5. *f(n)* = , *t* = 1 day n = 9295
6. *f(n)* = , *t* = 1 minute n = 8
7. Suppose we are comparing two sorting algorithms and that for all inputs of size the first algorithm runs in seconds, while the second algorithm runs in seconds. For which integer values of does the first algorithm beat the second algorithm? (4 points)

Explain in detail how you got your answer or paste code that solves the problem (2 point):

import math

for n in range(1, 10000):

    if ((4\*math.pow(n,3)) < (64\*n\*math.log2(n))):

        print(n)

        break

1. Give the complexity of the following methods. Choose the most appropriate notation from among , , and . (8 points each)

**int** function1(**int** n) {

**int** count = 0;

**for** (**int** i = n / 2; i <= n; i++) {

**for** (**int** j = 1; j <= n; j \*= 2) {

count++;

}

}

**return** count;

}

**Answer:**

**int** function2(**int** n) {

**int** count = 0;

**for** (**int** i = 1; i \* i \* i <= n; i++) {

count++;

}

**return** count;

}

**Answer:**

**int** function3(**int** n) {

**int** count = 0;

**for** (**int** i = 1; i <= n; i++) {

**for** (**int** j = 1; j <= n; j++) {

**for** (**int** k = 1; k <= n; k++) {

count++;

}

}

}

**return** count;

}

**Answer:**

**int** function4(**int** n) {

**int** count = 0;

**for** (**int** i = 1; i <= n; i++) {

**for** (**int** j = 1; j <= n; j++) {

count++;

**break**;

}

}

**return** count;

}

**Answer:**

**int** function5(**int** n) {

**int** count = 0;

**for** (**int** i = 1; i <= n; i++) {

count++;

}

**for** (**int** j = 1; j <= n; j++) {

count++;

}

**return** count;

}

**Answer:**