FALL 2019: **COT 6405** Analysis of Algorithms

[Homework 1; Suggest completion before next class Sep 4]

General submission guidelines and policies: Add the following signed statement. Without this statement, your homework will not be graded.

> I have adhered to the collaboration policy for this class. In other words, everything written down in this submission is my own work. For problems where I received any help, I have cited the source, and/or named the collaborator.

Read the handout on **Homework guidelines and collaboration policy** from your course website before you start on this homework. This is very important.

Problems

These are all background problems on prerequisite material you should know. They are all exercises and there is no need to submit any solutions.

1. (**Exercise**) [**Miscellaneous**]

   (a) How many bits would I need to write down an integer a in base b?

   (b) How many bits would I need to write down an integer n^2 in base n?

   (c) A tree can never be implemented using arrays. True/False? Why or why not?

2. (**Exercise**) [**Big-Oh Notation**]

   (a) In the box provided below, write down in big-Oh notation the time complexity of the following code/pseudocode:

Time Complexity of Test1 is

Algorithm 1 Test1()

```
1: for i = 1 to N do
2:     for j = i down to 1 do
3:         print i+j
4:     end for
5: end for
```

   (b) Prove the time complexity that you claimed in (a) above.

   (c) Prove that the time complexity of Test1 is not $O(N)$.

(d) Circle the right choice. To **prove** the claim that $f(n) = O(g(n))$, we need to:

i. prove the existence of at least one value of n that satisfies the big-Oh relationship

ii. prove the existence of two positive constants that satisfy the definition of the big-Oh relationship.

iii. prove by induction

iv. prove by contradition

(e) Circle the right choice. To **disprove** the claim that $f(n) = O(g(n))$, we need to:

i. prove the existence of at least one value of n that satisfies the big-Oh relationship

ii. prove the existence of two positive constants that satisfy the definition of the big-Oh relationship.

iii. prove by induction

iv. prove by contradition

(f) Prove or disprove the following claim:

$$16n = O(n^3)$$

(g) Prove or disprove the following claim:

$$n^3 = O(16n)$$

Mark the statements that are False for problems (h) through (r). No explanation is required for any of the problems.

(h) $3n(\log n)^2 + 4n = O(n(\log n)^2)$ for $c = 7$ and $n_0 = 1$.

(i) $n(\log n)^2 = O(n^2 \log n)$ for $c = 1$ and $n_0 = 1$.

(j) $n^2 \log n = O(2n^2 \log n + 1)$ for $c = 1$ and $n_0 = 1$.

(k) If $f(n) = O(g(n))$ and $g(n) = O(h(n))$, then

$$f(n) = O(h(n))$$

.

(l) If

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0,$$

then $f(n) = \Omega(g(n))$ cannot be true becaue $g(n)$ grows asymptotically faster than $f(n)$.

(m) $\frac{10}{9} > \frac{5}{4}$.

(n) For any $x > 0$,

$$\log_x 1 = 0$$

(o) $k = O(1)$ does not mean that $k = 1$.

(p) $k = O(n)$ is the same as saying $k = n$.

3. (**Exercise**) [**Kattis Submission**] The goal here is to learn how to make kattis submissions. Register with katiis using a login name that I can recognize (or let me know what your account ID is). Read the problem called `twostones`, which can be found at `https://open.kattis.com/problems/twostones`

   Prepare a programming solution in a language of your choice and learn how to submit it on kattis. The system will not accept your solution if your code has syntax errors or does not solve the problem correctly on its test data. Figure out how you can convince someone else that you have solved the problem (i.e., a screenshot of your profile, or a confirmation of acceptance of your submission.

4. (**Exercise**) Solve as many problems as possible from Exercises 3-2 and 3-3(a) on p61.

5. (**Exercise**) Solve as many problems as possible in Exercise 4.5-1 on p96.

6. (**Exercise**) Solve Exercise 4.5-3 on p96 using the master theorem. Make sure you are able to explain how you got the recurrence in the first place.

7. (**Exercise**) Explain in a couple of sentences why the correct recurrence for InsertionSort is
   $$T(N) \leq T(N-1) + O(N)$$
   Hint: Consider the last iteration.

8. (**Exercise**) Solve as many problems as possible from Exercises 4-1 and 4-3 on p107-108.