# 22C:31 Exam 1

## Due via ICON dropbox by 5 pm, Friday, Sept 25th

---

You should feel free to use your textbook and notes. However, you should not seek help on the internet or from any other person. By turning in your exam, you are confirming that you've completed your exam within these constraints. You may send me e-mail seeking clarifications; these I can provide, but I will not provide any hints. The exam is worth 150 points. This translates to 15% of your grade.

1. Here are two problems on running time analysis.

   (a) **20 points.** Compare the following pairs of functions in terms of how fast they grow asymptotically. In each case, say whether $f(n) = O(g(n))$, $f(n) = \Omega(g(n))$, or $f(n) = \Theta(g(n))$. You don't have to provide any justification for your answers.

   | | $f(n)$ | $g(n)$ |
   |---|---|---|
   | a. | $100n + \log n$ | $n + (\log n)^2$ |
   | b. | $\log n$ | $\log(n^2)$ |
   | c. | $\frac{n^2}{\log n}$ | $n(\log n)^2$ |
   | d. | $(\log n)^{\log n}$ | $\frac{n}{\log n}$ |
   | e. | $n^{1/2}$ | $(\log n)^5$ |
   | f. | $n2^n$ | $3^n$ |

   (b) **25 points.** Consider the following pseudocode.

   **while**($n >= 0$)**do**
       $m \leftarrow n$
       **while**($m >= 1$)**do**
           $m \leftarrow m/2$
       $n \leftarrow n - 1$

   Express the running time of this code fragment as a function of $n$. Specifically, come up with a function $f(n)$ such that the running time of the function is $\Theta(f(n))$. Prove your claim by showing that the running time of the code fragment is both $O(f(n))$ and $\Omega(f(n))$.

2. **35 points.** Suppose you are given a set $S = \{a_1, a_2, \ldots, a_n\}$ of tasks, where task $a_i$ requires $p_i$ units of processing time to complete, once it has started. You have one computer on which to run these tasks, and the computer can run only one task at a time. Let $c_i$ be the *completion time* of task $a_i$, that is, the time at which task $a_i$ completes processing. Your goal is to minimize average completion time, that is, to minimize $\frac{1}{n} \cdot \sum_{i=1}^{n} c_i$. For example, suppose there are two tasks $a_1$ and $a_2$ with $p_1 = 3$ and $p_2 = 5$ and consider the schedule in which $a_2$ runs first followed by $a_1$. Then $c_2 = 5$ and $c_1 = 8$, and the average completion time is $(5 + 8)/2 = 6.5$.

   (a) Give an algorithm that schedules the tasks so as to minimize the average completion time. Each task must run non-preemptively, that is, once task $a_i$ is started, it must run continuously for $p_i$ units of time. Prove that your algorithm minimizes the average completion time, and state the running time of your algorithm.

   (b) Suppose now that the tasks are not all available at once. That is, each task has a *release time* $r_i$ before which it is not available to proceed. Suppose that we allow *preemption*, so that a task can be suspended and restarted at a later time. For example, task $a_i$ with processing time $p_i = 6$ may start running at time 1 and be preempted at time 4. It can then resume at time 10 but be preempted at time 11 and finally resume at 13 and complete at 15. Task $a_i$ has run for a total of 6 time units, but its running time has been divided into three pieces. We say that the completion time of $a_i$ is 15. Give an algorithm that schedules the tasks so as to minimize average completion time in this new scenario. Prove that your algorithm minimizes the average completion time and state the running time of your algorithm.

3. **35 points.** Problem 8, pages 109-110, Chapter 3.
4. **35 points.** Problem 19, pages 198-199, Chapter 4. I do realize that this is a problem that also appears in Homework 2. I give this problem in the hope that you've already thought about it.

---