CS 31 Spring 2016
Midterm Exam

Ground rules

• This exam is due at 10:00 am on Friday, May 6.

• Until then, you may not discuss this exam with anyone other than Professor Cormen. Period. Not even with the TA or graders.

• The only written sources you may consult are your copy of *Introduction to Algorithms*, third edition; your own lecture notes; and homeworks and solutions from this term’s CS 31. Consulting any other source, either animate or inanimate, will be treated as a violation of the Academic Honor Principle.

• You should answer all the problems. The maximum score for this exam is 145 points. The points are not necessarily assigned to problems in proportion to difficulty. Unlike the homework assignments, you do not need to meet the 60% threshold to receive points for a problem.

• You may buy a hint for any problem (or more than one problem). The procedure is as follows. You visit Professor Cormen in his office and tell him how much you have already figured out about the problem. He might offer you one or more hints and tell you the price, in points, of each. You then decide which hint, if any, to buy. You will receive the hint, and the agreed-upon number of points will be deducted from your total for that problem. You will never be charged for a hint without your full knowledge and consent.

• Like the homework assignments, you must submit your exam as a PDF on the Canvas site. Start each problem on a separate page, and please include the problems in order of how they appear in this exam. Indicate the problem number for each problem.

• Any changes or clarifications will be announced via email.

Problem 1  (20 points)

The master theorem is well suited for recurrences of the form $T(n) = aT(n/b) + f(n)$. Consider a recurrence of the form

$$T(n) = \sum_{i=1}^{k} T(\alpha_i n) + f(n),$$

where each $\alpha_i$ is a constant fraction in the range $0 < \alpha_i < 1$. Such a recurrence falls under the umbrella of the master theorem only if all the $\alpha_i$ are equal. (That is, the $\alpha_i$ being equal is a necessary condition, but it is not sufficient.)

In the remainder of this problem, you will answer questions about the recurrence

$$T(n) = \sum_{i=1}^{k} T(\alpha_i n) + f(n),$$

where the following conditions hold:
• Each $\alpha_i$ is a constant fraction in the range $0 < \alpha_i < 1$. Do not assume that all the $\alpha_i$ are equal.

• $f$ is a function with the following properties:
  1. $f(x) > 0$ for all $x > 0$.
  2. $f(xy) = f(x) \cdot f(y)$ for all nonnegative $x$ and $y$. (Note that if $f(n) = n^c$ for some constant $c$, then this property holds, but do not assume that $f$ is of the form $n^c$.)

(a) (10 points)

Suppose that $\sum_{i=1}^{k} f(\alpha_i) < 1$. Give an asymptotically tight bound on $T(n)$. Prove that your answer is correct by using the substitution method. Don’t worry about the base case.

(b) (10 points)

Suppose that $\sum_{i=1}^{k} f(\alpha_i) = 1$. Give an asymptotically tight bound on $T(n)$. Prove that your answer is correct by using the substitution method. Don’t worry about the base case.

Problem 2 (10 points)

A new restaurant, Vern’s Tavern, opened up near my house recently. It took a little while for Vern’s Tavern to gain a clientele. The first time I went there, I was the only customer. The second time, there was just one other customer. The third time, there were two other customers. In fact, every time I went there, there was one more customer than the time before. Every time I went to Vern’s Tavern, I had a hat and so did all the other customers. (You are finding this story entirely plausible, are you not?)

Vern’s Tavern has a hat-retrieval professional who gives hats back to customers in a random order, and Vern’s policy is that all customers must give their hats to said hat-retrieval professional. Show that the expected number of times that I got my own hat back in my first $n$ visits to Vern’s Tavern is $\Theta(\lg n)$.

Problem 3 (40 points)

Consider the problem of making change for $n$ cents using the fewest number of coins. Assume that each coin’s value is an integer.

(a) (5 points)

Describe a greedy algorithm to make change consisting of quarters, dimes, nickels, and pennies. Prove that your algorithm yields an optimal solution, and state its running time.

(b) (10 points)

Suppose that the available coins are in the denominations that are powers of $c$, i.e., the denominations are $c^0, c^1, \ldots, c^k$ for some integers $c > 1$ and $k \geq 1$. Show that the greedy algorithm always yields an optimal solution.
(c) (5 points)
Give a set of coin denominations for which the greedy algorithm does not yield an optimal solution. Your set should include a penny so that there is a solution for every value of \( n \).

(d) (20 points)
Give an \( O(nk) \)-time algorithm that makes change for any set of \( k \) different coin denominations using the fewest number of coins, assuming that one of the coins is a penny.

**Problem 4 (20 points)**

Recall that a min-priority queue supports the operations \textsc{Insert}, \textsc{Extract-Min}, and \textsc{Decrease-Key}. You have seen how to implement a min-priority queue with a heap so that each operation takes \( O(\lg n) \) time, where \( n \) is the number of items in the min-priority queue.

Suppose that the keys are not arbitrary numbers but that they are integers in the range 0 to \( k \), inclusive. Assume that you know the value of \( k \) in advance.

(a) (5 points)
Describe how to implement a min-priority queue so that each of the above operations takes \( O(k) \) time.

(b) (10 points)
Now suppose that not only are the keys integers in the range 0 to \( k \), inclusive, but that the min-priority queue is being used in a problem for which the values returned by calls to \textsc{Extract-Min} are monotonically increasing over time. (In other words, if there are \( n \) calls to \textsc{Extract-Min}, let the \( i \)th call return the value \( x_i \). Then \( x_i \leq x_{i+1} \) for \( i = 1, 2, \ldots, n - 1 \).) Describe how to implement a min-priority queue so that any sequence of \( m \) operations takes a total of \( O(m + k) \) time.

(c) (5 points)
Now suppose that the keys are nonnegative integers that may be arbitrarily large, but that the min-priority queue is being used in a problem for which

1. the values returned by calls to \textsc{Extract-Min} are monotonically increasing over time, and
2. at any time, the difference between the smallest key in the queue and the largest key in the queue is at most \( k - 1 \). Again, assume that you know the value of \( k \) in advance.

Describe how to implement a min-priority queue so that any sequence of \( m \) operations takes a total of \( O(mk) \) time.

Here are a few things to bear in mind for this problem:

- A heap is one way, but by no means the only way, to implement a min-priority queue.
- A given key may appear in a priority queue multiple times.
- An object \textit{has} a key, but it is not correct to say that an object \textit{is} a key. When we call \textsc{Decrease-Key}, we are given an object and a new value to which we wish to reduce that object’s key value. In particular, we identify the object by a handle to it in the min-priority queue, rather than by its “old” key.
Problem 5  (10 points)

Consider a binary operator $\otimes$ that is associative, but not necessarily commutative, and that has the identity $e$. Suppose that a red-black tree $T$ has two additional attributes. One added attribute is $q$, and the sentinel is defined to have $T.nil.q = e$. The other attribute is $r$. Consider any node $x$, and let $x_1, x_2, \ldots, x_m$ be the nodes in the subtree rooted at $x$, listed according to an inorder walk of this subtree. Then we define the value of $x.r$ to be $x_1.q \otimes x_2.q \otimes \cdots \otimes x_m.q$.

Show how to update the $r$ attributes in $O(1)$ time after a left rotation and after a right rotation. Then give specific implementations of the $q$ and $r$ attributes, the operator $\otimes$, and the identity $e$ for maintaining the size attribute of order-statistic trees.

Problem 6  (25 points)

Consider an ordinary binary min-heap data structure with $n$ elements that supports the instructions INSERT and EXTRACT-MIN in $O(\log n)$ worst-case time.

(a) (20 points)

Give a potential function $\Phi$ such that the amortized cost of INSERT is $O(\log n)$ and the amortized cost of EXTRACT-MIN is $O(1)$, and show that your potential function yields these amortized time bounds.

(b) (5 points)

Professor Fifofum has spent his life trying to find a potential function for a binary min-heap such that the amortized cost of INSERT is $o(\log n)$ and the amortized cost of EXTRACT-MIN is $o(\log n)$. Explain why Professor Fifofum has wasted his life.

Problem 7  (20 points)

A rectangle is rectilinearly oriented if its sides are parallel to the $x$- and $y$-axes. Recall that two rectilinearly oriented rectangles overlap if and only if they overlap in both the $x$ and $y$ dimensions.

Describe an $O(n \log n)$-time algorithm that takes as input a set of $n$ rectilinearly oriented rectangles and produces as output either one pair of rectangles in the set that overlap or an indication that no rectangles in the set overlap. Assume that each rectangle is represented by its minimum and maximum $x$- and $y$-coordinates. The input might contain many overlapping rectangles, but the algorithm need output only one overlapping pair if there is any. Hint: Move a vertical line across the set of rectangles.