CPSC-320 Intermediate Algorithm Design and Analysis Winter 2000

Sample Midterm I Exam

You should try all three questions on the exam. Make sure you give yourself time for each part and remember that you will get partial credit if your work is on the right track.

You may refer to your lecture notes and the text during the exam. All material up to and including the lower bounds lecture is required for the exam. In the actual midterm, I will space the problems so that you can write the solutions in the midterm handout.

1. (a) Rank the following functions by order of growth; that is, find an arrangement $g_1, \ldots, g_7$ of the functions satisfying $g_1(n) = o(g_2(n))$, $g_2(n) = o(g_3(n))$, and so on. Partition your list into equivalence classes such that $f(n)$ and $g(n)$ are in the same class if and only if $f(n) = \Theta(g(n))$. (If you wish, you can assume all logs are to the base 2.)

$$n^3 - 10n, \quad 8^{\log n}, \quad n^n, \quad (3/2)^n, \quad \sum_{i=1}^{n} i \log i, \quad \sum_{i=1}^{n} (3i - 2).$$

   (b) State whether the following statement is true or false and give a brief explanation of your answer. Let $f(n)$ and $g(n)$ be non-negative functions defined over the non-negative integers. If $f(n) = o(g(n))$ then it must be the case that $g(n) \neq O(f(n))$.

2. Suppose S is a nonempty subset of the numbers in the range $[1, \ldots, n]$. We say that S is *independent* if and only if no pair of numbers in the subset are consecutive in the usual ordering. For example, if $n = 5$ then the subsets $\{1, 3, 5\}$, $\{2, 5\}$, and $\{4\}$ are all independent sets.

   (a) For the case $n = 5$, list *all* the possible independent subsets. (There are 12 of them.)

   (b) Let $I(n)$ be the number of independent subsets of numbers in the range $[1, \ldots, n]$. nodes. Give a recurrence relation for $I(n)$.
   (Suggestion: consider separately the independent sets containing the number n and those that don't contain the number n.)

   (c) Explain whether you think $I(n)$ is polynomial in n or exponential in n. (You don't need to solve the recurrence exactly to figure this out.)

3. A lucky number is any positive integer that passes through the following sieve: begin by removing every second number, then every third number from the remaining set; then remove every fourth number from the set left by the first two passes; and so forth. The first few lucky numbers are $1, 3, 7, 13, 19....$

   To find all lucky numbers smaller than n, we can use one of two algorithms. The first algorithm is a direct implementation of the definition: it uses a Boolean array of length n and makes repeated passes over the array, removing numbers until it completes a pass without removing any number. The second algorithm maintains a linked list of numbers still thought to be lucky and, at each pass, shrinks the list as needed, terminating when it completes a pass in which no number is removed.

   Analyze the worst-case behavior of these two algorithms as a function of n.