CPSC-320 Intermediate Algorithm Design and Analysis Winter 2000

Sample Midterm Exam I Solutions

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.)

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

Solution: We assume that logs are to the base 2. The order, starting from the slowest-growing function, is:

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

Note: the function $8^{\log_2 n}$ equals n^3. Therefore, the functions $n^3 - 10n = \Theta(8^{\log n})$.

(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 $g(n) \neq O(f(n))$.

Solution: True. Intuitively, if $f(n) = o(g(n))$ then $f(n)$ grows more slowly than $g(n)$. Therefore, $g(n)$ grows faster than $f(n)$ and so it cannot be the case that $g(n)$ is bounded by a constant times $f(n)$.

More precisely, since $f(n) = o(g(n))$ it must be the case that $\lim_{n\to\infty} f(n)/g(n) = 0$. Therefore for all constants c, there must be an integer N such that

$$f(n)/g(n) \leq 1/c \text{ for all } n \geq N.$$

Therefore, for all constants c, there is an integer N such that

$$g(n) \geq cf(n) \text{ for all } n \geq N.$$

Therefore, it cannot be the case that $g(n) = O(f(n))$ and so 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.)

Solution: These are: $\{1\}$, $\{2\}$, $\{3\}$, $\{4\}$, $\{5\}$, $\{1,3\}$, $\{1,4\}$, $\{1,5\}$, $\{2,4\}$, $\{2,5\}$, $\{3,5\}$, $\{1,3,5\}$.

(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.)

Solution: The recurrence is

$$I(n) = I(n-1) + I(n-2) + 1, n > 2; \quad I(1) = 1, I(0) = 0.$$

To see why the form for the general recurrence is true, note that any set which is an independent subset of $[1, 2, \ldots, n-1]$ is also an independent subset of $[1, 2, \ldots, n]$. There are $I(n-1)$ insependent subsets of $[1, 2, \ldots, n-1]$. The remaining independent subsets of $[1, 2, \ldots, n]$ must contain n. Any subset which is an independent subset of $[1, 2, \ldots, n-1]$, with n added in, is such a set. There are $I(n-2)$ such subsets. The only other possible subset containing n is the set consisting of just n, i.e. the set $\{n\}$. This account for the "+1" in the recurrence.

(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.)

Solution: From the recurrence, we can see that for $n > 2$, $I(n) \geq 2I(n-2)+1$. We can solve this using the iteration method to see that $I(n)$ is lower bounded by an exponential function in n. For simplicity, assume that n is even.

$$\begin{aligned}
I(n) &\geq 1 + 2I(n-2) \\
&\geq 1 + 2 + 4I(n-4) \\
&\geq 1 + 2 + 4 + 8I(n-6) \\
&\geq \ldots \\
&\geq 1 + 2 + 4 + \ldots 2^i + 2^{i+1}I(n-2i).
\end{aligned}$$

Let $i = n/2 - 1$ in the last line above. Then we have that

$$\begin{aligned}
I(n) &\geq 1 + 2 + 4 + \ldots 2^{n/2-1} + 2^{n/2}I(2) \\
&= 1 + 2 + 4 + \ldots 2^{n/2-1} + 2^{n/2+1} \\
&\geq 2^{n/2}.
\end{aligned}$$

The function $2^{n/2}$ is considered exponential in n since it is a constant (namely $2^{1/2}$) to the power n.

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.

Solution: The first algorithm takes time $\Theta(n)$ per pass, since on each pass the whole array must be scanned. How many passes are there? After the first round, about $n/2$ numbers remain. In the second round, about a third of remaining numbers are removed and $2/3$ remain. So the number remaining is $n/2x2/3 = n/3$. After the third round, $n/4$ are left, and so on. And so on, so that at the start of the ith round, n/i numbers remain.

No number is removed when $i > n/i$. The smallest i satisfying this inequality is just greater than $\sqrt{n}$. So, the number of rounds is about $\sqrt{n}$.

The total number of steps of the algorithm is therefore $\Theta(n^{3/2}$.

In the second algorithm, the linked list shrinks as numbers are removed. At the ith round, the size of the linked list is about n/i and so the number of steps is $\Theta(n/i)$. Adding up the costs of each phase, the total cost is

$$n + n/2 + n/3 + \ldots + n/\sqrt{n} = n(1 + 1/2 + 1/3 + \ldots 1/\sqrt{n}).$$

The sum $1+1/2+1/3+\ldots 1/\sqrt{n}$ is the harmonic sum and thus has value $\Theta(\log \sqrt{n}) = \Theta(\log n)$. Therefore, the total running time is $\Theta(n \log n)$.