CS 173, Spring 2008
Midterm 2 Solutions

Problem 1: True/false (12 points)

Label each of the following statements as true or false.

(a) $\exists x \in \mathbb{Z}, \forall y \in \mathbb{Z}, x < y$

Solution: False. The statement claims that there is one particular integer x which is smaller than all integers (including itself).

(b) $n + \log_2 n$ is $O(n)$

Solution: True. As n gets large, $\log_2 n$ becomes very small compared to n so we can ignore that term.

(c) $n!$ is $O(2^n)$

Solution: False. First, we showed in lecture 16 that $2^n < n!$ when $n \geq 4$. Second, as n gets large, the $n!$ increases at a rate proportional to n whereas 2^n increases only by a factor of 2. This is perhaps the best way to remember which way this inequality goes.

(d) $n \log_8 n$ is $\Theta(n \log_2 n)$

Solution: True. To change the base of a log, you multiply by a fixed constant. By "constant", I mean a number which depends on the two bases (2 and 8) but not on the input n. Constant multipliers don't change big-O growth.

(e) The function $g : \mathbb{Z} \to \mathbb{R}$ defined by $g(x) = x + 1$ is onto.

Solution: False. How can it be onto, since there are a lot fewer integer inputs than there are real number outputs? Or, said another way, the output of g is always an integer, so non-integer values (e.g. 3.1415) can never be outputs of g.

Problem 2: Short answer (9 points)

(a) The number of ways to pick a k-element subset from a set containing n elements is written $C(n, k)$ or $\binom{n}{k}$. Give an equation for computing this quantity using factorials.

Solution:

$$\frac{n!}{k!(n-k)!}$$

BTW, this won't be the last time someone expects you to remember this particular formula. It's a good one to get memorized.

(b) Let $f : \mathbb{N} \to \mathbb{Z}$ be defined by $f(x) = 7x + 12$. Prove that f is one-to-one.

Solution: Let x and y be two natural numbers and suppose that $f(x) = f(y)$.

Since $f(x) = f(y)$, $7x + 12 = 7y + 12$ by the definition of f. So, by high school algebra, $7x = 7y$ and therefore $x = y$.

So, we've shown if $f(x) = f(y)$ then $x = y$. This means that f is one-to-one.

Problem 3: Recursive definition (10 points)

(a) Here is a recursive definition of a set S, which contains pairs of numbers:

1) $(2, 1) \in S$ and $(1, 1) \in S$
2) If $(x, y) \in S$, then $(xy, 1) \in S$
3) If $(x, y) \in S$ and $(p, q) \in S$, then $(x, p) \in S$.

Give a non-recursive definition for the set S. Explain briefly and/or show your work.

Solution: Since $(2, 1) \in S$ and $(2, 1) \in S$, $(2, 2) \in S$ by (3).

Since $(1, 1) \in S$ and $(2, 1) \in S$, $(1, 2) \in S$ by (3).

Since $(2, 2) \in S$, then $(4, 1) \in S$ by (2). So if we feed $(4, 1)$ and $(2, 1)$ to (3), we find that $(4, 2) \in S$. If we repeat this process, we can keep multiplying the first coordinate by 2. So S must contain every pair of the form $(2^n, 2)$ or $(2^n, 1)$, where $n \geq 0$.

But, if $(2^n, 2)$ and $(2^m, 2)$ are in S, then (3) implies that $(2^n, 2^m)$ is in S.

So S is the set of all pairs of the form $(2^n, 2^m)$, where $m, n \geq 0$.

(b) Find a closed-form solution for the following recurrence relation with the given initial condition. A closed-form solution is a function $T(n)$ that yields that same values as the recurrence relation but is non-recursive. You should be able to find the solution by *unrolling* the recurrence and then applying a formula you have seen before to find a closed form for a summation. Show your work.

$T(n) = T(n - 1) + 2n$ with initial condition $T(0) = 0$

Solution:

$$\begin{aligned}
T(n) &= T(n-1) + 2n \\
&= T(n-2) + 2(n-1) + 2n \\
&= T(n-3) + 2(n-2) + 2(n-1) + 2n \\
&= T(n-4) + 2(n-3) + 2(n-2) + 2(n-1) + 2n \\
&= T(0) + 2(1 + 2 + \ldots + n) \\
&= 2(1 + 2 + \ldots + n) \\
&= 2\frac{n(n+1)}{2} \\
&= n(n+1)
\end{aligned}$$

Problem 4: Algorithms (9 points)

Clarification given at exam: assume the input list a_1,...,a_n is sorted in decreasing order.

The following algorithm is a recursive form of binary search. It takes as input an arbitrary list of n real numbers a_1,...,a_n and determines if a given number x is in the list. If the number is in the list, the function returns the position of x, if x is not in the list it returns the value 0. In the following code i and j are integers indicating the current search range (i.e. $i = 6$ and $j = 10$ means a_6 through a_{10} are to be searched).

(a) Fill in the 2 blank lines below with pseudo-code that will correctly execute binary search recursively.

```
procedure binary search($x$,$i$,$j$,$a_1$,...,$a_n$)
$m := \lfloor (i+j)/2 \rfloor$
if ($x = a_m$) then
    $location := m$
else if ($x > a_m$ and $i < m$) then
    SOLUTION: binary search($x, i, m-1, a_1$,...,$a_n$)
else if ($x < a_m$ and $m < j$) then
    SOLUTION: binary search($x, m+1, j, a_1$,...,$a_n$)
else $location := 0$
```

Commentary on solution: The first recursive call needs to inspect the half of the list containing the larger numbers, because it happens in the case where the middle element of the list a_m was smaller than the target value x. Since the condition checks that $i < m$, this has to be the first half of the list. This is why the input list needs to have been in decreasing order.

The pseudo-code in the book (p. 314) assumes increasing sorted order. Switching to decreasing was unintentional on our part and probably made this problem a bit more tricky than we had planned.

(b) Suppose the function $T(n)$ returns the number of operations needed by binary search to find x (in the worst case) in a list of length n. Write a recurrence relation (recursive formula) for $T(n)$. Your recurrence should count the comparison and arithmetic operations done by binary search, but it need not be exact: you can use letters such as c in your formula to represent constant numbers.

Solution: $T(n) = T(\frac{n}{2}) + c$. Each run of the binary search function does one recursive call, on a problem half the size of the original. It also does some constant amount of work, e.g. doing the tests for the if/then statements, computing the value m.

Problem 5: Induction (10 points)

Let's define a sequence of numbers x_n as follows:

Base: $x_1 = 1$, $x_2 = 7$

Induction: for every $n \geq 2$, $x_{n+1} = 7x_n - 12x_{n-1}$

Use induction to prove that $x_i = 4^n - 3^n$ for every integer $n \geq 1$. Hint: the algebra in the inductive step should work out easily.

Clarification posted during exam: You need to prove $x_n = 4^n - 3^n$, not $x_i = 4^n - 3^n$.

Solution: Proof by induction on n.

Base: If $n = 1$ then $4^n - 3^n = 4 - 3 = 1$ and x_1 is defined to be 1.

If $n = 2$ then $4^n - 3^n = 16 - 9 = 7$ and x_2 is defined to be 7.

Induction: Suppose that $x_k = 4^k - 3^k$ for every k in the range $[1, n]$, where $n \geq 2$. We need to show that $x_{n+1} = 4^{n+1} - 3^{n+1}$.

x_{n+1} is defined to be $7x_n - 12x_{n-1}$. By the inductive hypothesis $x_n = 4^n - 3^n$ and $x_{n-1} = 4^{n-1} - 3^{n-1}$. Substituting in these values, we get that

$$\begin{aligned}
x_{n+1} &= 7x_n - 12x_{n-1} \\
&= 7(4^n - 3^n) - 12(4^{n-1} - 3^{n-1}) \\
&= 7 \cdot 4^n - 7 \cdot 3^n - 12 \cdot 4^{n-1} + 12 \cdot 3^{n-1} \\
&= 7 \cdot 4^n - 7 \cdot 3^n - 3 \cdot 4^n + 4 \cdot 3^n \\
&= 7 \cdot 4^n - 7 \cdot 3^n - 3 \cdot 4^n + 4 \cdot 3^n \\
&= (7 - 3) \cdot 4^n - (7 - 4) \cdot 3^n \\
&= 4 \cdot 4^n - 3 \cdot 3^n \\
&= 4^{n+1} - 3^{n+1}
\end{aligned}$$

So $x_{n+1} = 4^{n+1} - 3^{n+1}$, which is what we needed to show.

Commentary on solution: The hard part is setting up the assumptions and goal of the inductive step. If you've done that right, the algebra in the inductive step is fairly straightforward. (Or, if it didn't work out due to exam pressure, you'll still get a lot of partial credit.)

You need to use a "strong" hypothesis at the start of the inductive step, because the inductive step needs to refer back to results for two previous values: n and $n - 1$. This is also why we needed two base cases.