CSI 2101 Discrete Structures | Winter 2012
Prof. Lucia Moura | University of Ottawa

**Homework Assignment #3** (100 points, weight 5%)
Due: Thursday, March 22, at 1:00pm (in lecture)

---

**Induction and program correctness**

1. (20 points) **Mathematical Induction**
   Use induction to prove that for every positive integer $n$,

$$\sum_{k=1}^{n} k2^k = (n-1)2^{n+1} + 2.$$

Let $P(n)$ be the statement "$\sum_{k=1}^{n} k2^k = (n-1)2^{n+1} + 2$." We show that $P(n)$ is true for all positive integers $n$.

**Basis step:** We show that $P(1)$ is true. On the left hand side, we have:

$$\sum_{k=1}^{1} k2^k = 1 \cdot 2^1 = 2.$$

On the right hand side, we have:

$$(1-1)2^{1+1} + 2 = 2.$$

Thus, $P(1)$ is true.

**Inductive hypothesis:** Assume that for a positive integer $m$, we have that $P(m)$ is true, which gives:

$$\sum_{k=1}^{m} k2^k = (m-1)2^{m+1} + 2.$$

**Inductive step:** We have that:

$$\begin{aligned}
\sum_{k=1}^{m+1} k2^k &= (m+1)2^{m+1} + \sum_{k=1}^{m} k2^k \\
&= (m+1)2^{m+1} + (m-1)2^{m+1} + 2 && \text{by inductive hypothesis} \\
&= (m+1+m-1)2^{m+1} + 2 \\
&= (2m)2^{m+1} + 2 \\
&= m2^{m+2} + 2 \\
&= ((m+1)-1)2^{(m+1)+1} + 2
\end{aligned}$$

Thus, when $P(m)$ is true, we have that $P(m+1)$ is true. Since the base case $P(1)$ is also true, we have that $P(n)$ holds for all positive integers $n$.

2. (25 points) **Strong induction**

   Use strong induction to show that every positive integer $n$ can be written as a sum of distinct powers of two, that is, as a sum of the integers $2^0 = 1, 2^1 = 2, 2^2 = 4, 2^3 = 8$, and so on.

   Hint: for the inductive step, separately consider the case where $k+1$ is even and where it is odd. When it is even, note that $(k+1)/2$ is an integer.

   Let $P(n)$ be the claim that $n$ can be written as a sum of distinct powers of two. We show that $P(n)$ is true for all positive integers $n$.

   **Basis step:** As $1 = 2^0$, we have that $P(1)$ is true.

   **Inductive hypothesis:** Assume for a positive integer $k$ that $P(i)$ is true for all $1 \leq i \leq k$.

   **Inductive step:** We consider two cases, namely when $k+1$ is even and when $k+1$ is odd. If $k+1$ is even, then $(k+1)/2$ is an integer, and by the inductive hypothesis, we can express $(k+1)/2$ by a sum of distinct powers of two. We can then multiply this sum by 2, which simply increases the exponent of each power of two by 1, so this is again a sum of distinct powers of two that is equal to $k+1$.

   When $k+1$ is odd, we have that $k$ is even. By the inductive hypothesis, we can express $k$ as a sum of distinct powers of two. However, since $k$ is even, the sum cannot contain $2^0 = 1$. Thus, we can add $2^0$ to this sum, which remains a sum of distinct powers of two, and equals $k+1$.

   Thus, in both cases, we can express $k+1$ as a sum of distinct powers of two, so when $P(i)$ is true for all $1 \leq i \leq k$, we have that $P(k+1)$ is true. Since $P(1)$ is true, this means that the claim is true for all positive integers, as show by strong induction.

3. (25 points) **Structural Induction**

   (a) Give a recursive definition of the function $ones(s)$, which counts the number of ones in a bit string $s$ (a bitstring is a string over the alphabet $\Sigma = \{0, 1\}$).

   (b) Use structural induction to prove that $ones(s \cdot t) = ones(s) + ones(t)$; where the symbol "$\cdot$" denotes concatenation of strings.

   Hint: in some of your steps you need to rely on the recursive definition of strings and of concatenation given in the textbook, as well as on the definition of $ones(s)$ given by you.

   Take $\Sigma = \{0, 1\}$. We define $ones(\lambda) = 0$, and $ones(wx) = ones(w) + x$ for $w \in \Sigma^*$, $x \in \Sigma$.

Let $P(t)$ be the claim that for any $s \in \Sigma^*$, $ones(st) = ones(s) + ones(t)$. We wish to show that $P(t)$ holds for $t \in \Sigma^*$. We demonstrate this using structural induction, using the definitions of strings and concatenation.

**Basis step:** Consider $P(\lambda)$:

$$\begin{aligned} ones(s\lambda) &= ones(s) && \text{(definition of strings)} \\ &= ones(s) + 0 \\ &= ones(s) + ones(\lambda) && \text{(definition of } ones\text{)} \end{aligned}$$

This concludes the base case.

**Inductive hypothesis:** Assume $P(t)$ is true, i.e. for any $s \in \Sigma^*$, $ones(st) = ones(s) + ones(t)$.

**Inductive step:** For $x \in \Sigma$, we have that:

$$\begin{aligned} ones(s(tx)) &= ones((st)x) && \text{(definition of concatenation)} \\ &= ones(st) + x && \text{(definition of } ones\text{)} \\ &= ones(s) + ones(t) + x && \text{(inductive hypothesis)} \\ &= ones(s) + ones(tx) && \text{(definition of } ones\text{)} \end{aligned}$$

Thus, if $P(t)$ is true, then $P(tx)$ is true, i.e. the statement is true for the extension of $t$.

Thus, the claim holds by structural induction.

4. (30 points) **Correctness of recursive algorithms**
Prove that Algorithm 6 (recursive binary search algorithm) in page 314 (Section 4.4) is correct, as follows. Consider the following statement:

> $P(k)$ : " If $n$ is an integer and $a_1, a_2, \ldots, a_n$ are integers in increasing order, and $i, j, x$ are integers such that $1 \leq i \leq n, 1 \leq j \leq n$ and $j - i = k$, then procedure $binarysearch(i, j, x)$ calculates $location$, where $location = 0$ if there exists no $l$, $i \leq l \leq j$, with $a_l = x$, or $location = m$ and $a_m = x$ with $i \leq m \leq j$, otherwise."

Use strong induction to prove that $P(k)$ is true for all $k \geq 0$.

We will use strong induction to prove that $P(k)$ holds for all $k \geq 0$.

**Basis step:** If $k = 0$, then $i = j$, i.e. we are working with a sublist of length 1. Then by the algorithm, $m = i = j$, and we check to see if $a_m = x$. If this is the case, the algorithm sets $location$ to $m$ as required. If not, the recursive calls are not performed as $i \not< m$ and $j \not> m$, so the *else* statement is executed, and thus *location* is set to 0 as required.

**Inductive step:** Assume that $P(k)$ holds for $0, \ldots, k$ with $k \geq 0$.

Now consider the case for $k+1$. The algorithm begins by computing:

$$m = \lfloor \frac{i+j}{2} \rfloor$$

If $x = a_m$, then location is set to $m$ as required, and the algorithm terminates as no recursive calls are made. Otherwise, we have two cases, namely:

1. $x < a_m$. Thus, if $x$ appears in the sublist $a_i, \ldots a_j$, it must appear in the sublist $a_i, \ldots, a_{m-1}$.

   If $i < m$, then the first *else if* statement is executed, so the algorithm is called recursively with the sublist between $i$ and $m-1$. As $(m-1)-i < j-i = k+1$, we have that $(m-1)-i \leq k$. Furthermore, as $m > i$, then $m-i > 0$, or $(m-1)-i \geq 0$ so putting these two inequalities together, the recursive call executes on a list of length $(m-1)-i$ where $0 \leq (m-1)-i \leq k$. Thus, by the inductive hypothesis, this call correctly sets the value of *location* for $x$ in the sublist between $i$ and $m-1$.

   If, instead, $i \geq m$, we have that $i = m$ as $m$ falls between $i$ and $j$. Then there are no smaller elements in the sublist between $i$ and $j$ to check for $x$, and so $x$ does not appear in this sublist. The *else* statement is then executed, setting *location* to 0 as required.

2. $x > a_m$. Thus, if $x$ appears in the sublist $a_i, \ldots a_j$, it must appear in the sublist $a_{m+1}, \ldots a_j$.

   It is not possible that $j \leq m$, in this case, as $j \leq m$ means $j \leq \lfloor (i+j)/2 \rfloor$. We can drop the floor here, so $j \leq (i+j)/2$. This gives that $j \leq i$, or $j = i$. This implies $k+1 = 0$, which contradicts $k \geq 0$. Hence, in this case, the *else if* statement is always executed for the sublist between $m+1$ and $j$. We have that $j-(m+1) < j-i = k+1$, so $j-(m+1) < k$. Also, as $j > m$, $j-m > 0$, i.e. $j-m-1 \geq 0$, or $j-(m+1) \geq 0$. Thus, $0 \leq j-(m+1) \leq k$, so by the inductive hypothesis, the recursive call to binary search on the sublist from $m+1$ to $j$ correctly sets *location* to the required value.