Homework 2

SJSU Students

March 12, 2006

Problem 1 (5.1)

Let X_i be the indicator random variable denoting the event that the *counter* increases due to the INCREMENT operation and i = *1, 2, . . . , n*. Thus,

$$X_i = \begin{cases} 1 \text{ if the } \textit{counter} \text{ increases due to the } \textit{ith} \text{ INCREMENT operation} \\ 0 \text{ otherwise} \end{cases}$$

Let X be a random variable denoting the value of the *counter* after n INCREMENT operations. Then,

$$X = X_1 + X_2 + \ldots + X_n \tag{1}$$

By linearity of expectation, we have

$$E[X] = E[X_1 + X_2 + \ldots + X_n] = E[\sum_{i=1}^{n} X_i] = \sum_{i=1}^{n} E[X_i] \tag{2}$$

By Lemma 5.1, we have:

$E[X_i] = \Pr\{$*counter* increases after *ith* INCREMENT operation$\}$.

Suppose, before *ith* INCREMENT operation, *counter* is equal to n_i. If the *counter* increases after *ith* INCREMENT operation, its value is $n_{i+1} - n_i$ and the probability of such increase is $\frac{1}{(n_{i+1}-n_i)}$.

Thus,

$$E[X_i] = (n_{i+1} - n_i) \times \frac{1}{(n_{i+1} - n_i)} = 1. \tag{3}$$

and

$$E[X] = E[X_1 + X_2 + \ldots + X_n] = E[\sum_{i=1}^{n} X_i] = \sum_{i=1}^{n} E[X_i] = \sum_{i=1}^{n} 1 = n \qquad (4)$$

Thus, the expected value represented by the *counter* after n INCREMENT operations is exactly n.

Part b

Let X_i denote the increase in *counter* due to *ith* INCREMENT operation and i = *1, 2, . . . , n*. And let X be a random variable denoting the value of the *counter* after n INCREMENT operations. Then,

$$Var[X] = Var[X_1] + Var[X_2] + \ldots + Var[X_n]. \qquad (5)$$

Also,

$$Var[X_i] = E[X_i^2] - E^2[X_i] \qquad (6)$$

Since $n_i = 100i$, then

$$n_{i+1} - n_i = 100(i + 1) - 100i = 100. \qquad (7)$$

and we have:
Also,

$$Var[X_i] = E[X_i^2] - E^2[X_i] = (100^2 \times \frac{1}{100}) - 1^2 = 100 - 1 = 99. \qquad (8)$$

$$Var[X] = Var[X_1] + Var[X_2] + \ldots + Var[X_n] = 99n. \qquad (9)$$

Problem 2 (27.2-2) *Prove that a comparison network N with n inputs correctly sorts the input sequence $\langle n, n-1, \ldots, 1\rangle$ if and only if it correctly sorts the $n-1$ zero-one sequences $\langle 1, 0, 0, \ldots, 0\rangle, \langle 1, 1, 0, \ldots, \rangle, \ldots, \langle 1, 1, 1, \ldots, 1, 0\rangle$.*

First, suppose a comparison network sorts the input sequence $\langle n, n-1, \ldots, 1\rangle$ correctly. To see that it sorts the sequence $\langle \overbrace{1, \ldots, 1}^{k}, \overbrace{0, \ldots, 0}^{n-k}\rangle$, let $f_k(x)$ be the function which outputs 1 if $x - k > 0$ and outputs 0 otherwise. Notice this function is nondecreasing, so we can apply Lemma 27.1 from the book. This and the definition of f_k tells use that the following string of equalities hold:

$$\begin{aligned}
N(\langle \overbrace{1, \ldots, 1}^{k}, \overbrace{0, \ldots, 0}^{n-k}\rangle) &= N(f_k(\langle n, n-1, \ldots, 1\rangle)) \\
&= f_k(N(\langle n, n-1, \ldots, 1\rangle)) \\
&= \langle f_k(1), f_k(2), \ldots, f_k(n-1), f_k(n)\rangle \\
&= \langle \overbrace{0, \ldots, 0}^{n-k}, \overbrace{1, \ldots, 1}^{k}\rangle
\end{aligned}$$

So N sorts $\langle \overbrace{1, \ldots, 1}^{k}, \overbrace{0, \ldots, 0}^{n-k}\rangle$ and as this argument works for each k from 1 to n, each of the above zero-one sequences will be sorted.

On the other hand, suppose N does not sort $\langle n, n-1, \ldots, 1\rangle$. Then for some values $j < k$, the network N on input $\langle n, n-1, \ldots, 1\rangle$ swaps the order of j and k in the output. That is, N maps j to some position j_N and k to the position k_N and $j_N > k_N$. But this means by Lemma 27.1, that N maps $0 = f_j(j)$ to j_N and $f_j(k) = 1$ to k_N. So N fails to sort

$$\langle f_j(1), f_j(2), \ldots, f_j(n-1), f_j(n)\rangle = \langle \overbrace{1, \ldots, 1}^{j}, \overbrace{0, \ldots, 0}^{n-j}\rangle.$$

Hence, if N fails to sort $\langle n, n-1, \ldots, 1\rangle$ it fails to sort at least one of the sequences $\langle 1, 0, 0, \ldots, 0\rangle, \langle 1, 1, 0, \ldots, \rangle, \ldots, \langle 1, 1, 1, \ldots, 1, 0\rangle$. In other words, N sorts $\langle n, n-1, \ldots, 1\rangle$ only if it sorts each of

$$\langle 1, 0, 0, \ldots, 0\rangle, \langle 1, 1, 0, \ldots, \rangle, \ldots, \langle 1, 1, 1, \ldots, 1, 0\rangle.$$

This completes the proof of the if and only.

Problem 3 (27.3-5) *Consider two sequences of 0's and 1's. Prove that if evey element in one sequence is at least as small as every element in the other sequence, then one of the two sequences is clean.*

There are three cases for this problem and for each case at least one of the sequences is clean:

- The first sequence contains all zero's, which allow second sequence to contain 0's and 1's, in this case the first sequence is clean (all zeros).
- The first sequence contains mix of zero's and ones, which force the second sequence to contain only ones (for the condition to be satisfied), in this case the second sequence is clean (all ones).
- The first sequence contains all ones, which force the second sequence to contain only 1's, in this case both sequences are clean (all ones).

Problem 4 (27.5-3) *Suppose we have $2n$ elements $\langle a_1, a_2, ..., a_{2n} \rangle$ and wish to partition them into the n smallest and the n largest. Prove that we can do this in constant additional depth after separately sorting $\langle a_1, a_2, ..., a_n \rangle$ and $\langle a_{n+1}, a_{n+2}, ..., a_{2n} \rangle$.*

After separately sorting each list , we can have the n smallest and the n largest by doing the following: by applying the list as and input to a modified half cleaner, all the n smallest elements will be in one n sequence and n largest in the other. This method is equivalent to reverse the order of the second sorted sequence and do the half cleaner method after. Both these methods work because by doing so we are comparing the smallest elements of the first sequence with the largest elements of second sequence and the largest elements of the first sequence with the smallest elements of the second sequence. Thus all of the smallest elements are pushed up to the upper sequence and the largest elements are pushed down to the lower sequence.