CS21001 Discrete Structures, Autumn 2007

Class test 2: November 07, 2007 (7:00–8:00pm), Total Marks: 30

Roll No: _____ **Name:** _____

[*Write your answers in the respective spaces provided in the question paper itself. Answer briefly.*]

1 On two sides of a battle-field, there are two camps of soldiers. Initially, Camp 1 consists of n_1 soldiers, and Camp 2 of n_2 soldiers. Each camp sends one soldier to the battle-field. The two soldiers fight for one minute and exactly one of them dies. The living soldiers rejoins his camp. This process continues as long as each camp has (at least one) soldier to continue the battle. In terms of a C program, the battle looks as follows.

```
int battle ( unsigned int n1, unsigned int n2 )
{
   int t = 0, whowins;
   while ((n1 > 0) && (n2 > 0)) {         /* as long as both camps are non-empty */
      whowins = findWinner();                    /* findWinner() returns 1 or 2 */
      if (whowins == 1) --n2; else --n1;            /* the losing soldier dies */
      ++t;                                        /* increase time by 1 minute */
   }
   return t;                          /* return the total duration of the battle */
}
```

(a) Let m be the minimum possible duration (in minutes) of the battle, i.e., the minimum value returned by the function **`battle(n1,n2)`**. Determine m (in terms of n_1, n_2). Justify your answer (i.e., argue that no battle can last shorter than m minutes and that a battle can last for m minutes.) **(5)**

Solution In each iteration of the **`while`** loop, either n_1 or n_2 (but not both) decreases, i.e., after each fight (lasting for 1 minute), one soldiers dies. Therefore, in less than $\min(n_1, n_2)$ steps, both the camps remain non-empty and the battle continues, i.e., $m \geqslant \min(n_1, n_2)$.

On the other hand, in the case that all the soldiers of the smaller group die and nobody from the larger group dies, the battle lasts for $\min(n_1, n_2)$ minutes, i.e., $m \leqslant \min(n_1, n_2)$.

Thus, $m =$ $\min(n_1, n_2)$.

(b) Let M be the maximum possible duration (in minutes) of the battle. Determine M with justification. **(5)**

Solution We assume that n_1, n_2 are positive. After the end of the battle, both the camps cannot be empty simultaneously, since inside the loop, exactly one of n_1 and n_2 decreases, and the loop body is executed when *both* n_1 and n_2 are positive. Thus, after the end of the battle, at least one soldier remains alive, i.e., at most $n_1 + n_2 - 1$ soldiers die in the battle. Since each step (fight) involves the death of a soldier, $M \leqslant n_1 + n_2 - 1$.

Solution (*cond.*) On the other hand, in the case that $n_1 - 1$ soldiers first die from Camp 1 and then the last soldier from Camp 1 kills all of the n_2 soldiers from Camp 2, the battle lasts for exactly $(n_1 - 1) + n_2$ minutes, i.e., $M \geqslant n_1 + n_2 - 1$.

Thus, $M = \underline{\qquad n_1 + n_2 - 1 \qquad}$.

(c) Now imagine a modified situation in which two soldiers fight for one minute, but nobody dies. In that case, the two soldiers stop the fight and decide to go together to one of the two camps. This means that one of the two soldiers defects from his own camp and joins the other camp. This defection operation takes an additional minute (after the one-minute fight). Thus, each fight now has three outcomes: (a) the soldier from Camp 1 dies, (b) the soldier from Camp 2 dies, and (3) one of the two soldiers defect. The first two outcomes take 1 minute each, whereas the third outcome takes 2 minutes. Assume that if a soldier defects to the rival camp, he can never go back to his original camp.

Let M' be the maximum possible duration (in minutes) of the battle in this modified scenario. Determine M' with justification. You may assume that $n_1, n_2 \geqslant 2$. **(5)**

Solution A step involving defection takes a time of 2 minutes and does not involve the death of a soldier. Therefore, if all of the $n_1 + n_2$ soldiers defect, a total of $2(n_1 + n_2)$ minutes are spent. After that, at most $n_1 + n_2 - 1$ deaths are possible before the battle stops (as argued in Part (b)). Therefore, $M' \leqslant 2(n_1 + n_2) + (n_1 + n_2 - 1) = 3(n_1 + n_2) - 1$.

On the other hand, imagine a situation where all soldiers first defect. Except in the case $n_1 = n_2 = 1$, this defection is possible without each side being empty at any intermediate instant. Such a defection essentially changes Camp 1 to the original Camp 2 and Camp 2 to the original Camp 1. This is, in reality, a funny situation, but is a mathematical possibility. Now as argued in Part (b), $n_2 - 1$ soldiers from the modified Camp 1 dies, and finally the last soldier from the modified Camp 1 kills all of the n_1 soldiers from the modified Camp 2. In this situation, the battle lasts for $2(n_1 + n_2) + (n_2 - 1) + n_1$ minutes, i.e., $M' \geqslant 3(n_1 + n_2) - 1$.

Thus, $M' = \underline{\qquad 3(n_1 + n_2) - 1 \qquad}$.

2 Let the sequence $a_0, a_1, a_2, \ldots$ be defined recursively as follows.

$$\begin{aligned} a_0 &= 0, \\ a_n &= a_{n-1} + a_{n-2} + \cdots + a_1 + a_0 + 2^{n+1} \quad \text{for } n \geqslant 1. \end{aligned}$$

Deduce a closed-form formula for a_n for all $n \geqslant 0$. You may use any method that you find convenient. You may use the back of this page, if your calculations do not fit in this page. **(15)**

Solution With a little manipulation, we can convert the given recurrence relation to a more convenient form. For $n-1 \geqslant 1$, i.e., for $n \geqslant 2$, we have $a_{n-1} = a_{n-2}+a_{n-3}+\cdots+a_1+a_0+2^n$. That is, $a_n - a_{n-1} = a_{n-1} + 2^n$, i.e., $a_n = 2a_{n-1} + 2^n$. It is important to highlight that this simplified recurrence does not hold for $n = 1$. Indeed, we compute $a_1 = a_0+2^{1+1} = 0+4 = 4$. Thus, the modified recurrence relation (equivalent to the original one) looks as follows.

$$\begin{aligned} a_0 &= 0, \\ a_1 &= 4, \\ a_n &= 2a_{n-1} + 2^n \quad \text{for } n \geqslant 2. \end{aligned}$$

Here are several ways of solving this (or the original) recurrence relation.

Method 1: By unfolding the modified recurrence relation.

For $n \geqslant 1$, we have

$$\begin{aligned} a_n &= 2a_{n-1} + 2^n \\ &= 2(2a_{n-2} + 2^{n-1}) + 2^n = 2^2 a_{n-2} + 2 \times 2^n \\ &= 2^2(2a_{n-3} + 2^{n-2}) + 2 \times 2^n = 2^3 a_{n-3} + 3 \times 2^n \\ &= \cdots \\ &= 2^{n-1} a_1 + (n-1) \times 2^n = 2^{n-1} \times 4 + (n-1) \times 2^n \\ &= (n+1)2^n. \end{aligned}$$

Therefore, $a_n = \begin{cases} 0 & \text{if } n = 0, \\ (n+1)2^n & \text{if } n \geqslant 1. \end{cases}$

Method 2: Using the master theorem on the modified recurrence relation.

The characteristic equation of the modified recurrence relation is $x - 2 = 0$ which has only one simple root $x = 2$. Therefore, $a_n = b_n + c_n$ for $n \geqslant 1$, where the general solution is $b_n = u2^n$ and the particular solution is $c_n = vn2^n$ for some real constants u, v. We first determine v by putting $a_n = c_n$, i.e., $vn2^n = 2v(n-1)2^{n-1} + 2^n$, i.e., $vn = v(n-1) + 1$, i.e., $v = 1$. Therefore, $a_n = u2^n + n2^n$ for all $n \geqslant 1$. Putting $n = 1$ gives $a_1 = 4 = 2u + 2$, i.e., $u = 1$. (Note that in this case we must not determine u from a_0, since the modified recurrence relation does not hold for $n = 1$.) Thus, $a_n = \begin{cases} 0 & \text{if } n = 0, \\ (n+1)2^n & \text{if } n \geqslant 1. \end{cases}$

Method 3: Using generating functions on the modified recurrence relation.

Let $a(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n + \cdots$. Using the modified recurrence relation for $n \geqslant 2$ gives

$$\begin{aligned} a(x) &= a_0 + a_1 x + (2a_1 + 2^2)x^2 + (2a_2 + 2^3)x^3 + \cdots + (2a_{n-1} + 2^n)x^n + \cdots \\ &= a_0 + a_1 x + 2x(a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1} + \cdots) + \\ &\quad 2^2 x^2 (1 + (2x) + (2x)^2 + \cdots + (2x)^{n-2} + \cdots) \\ &= a_0 + a_1 x + 2x(-a_0 + a(x)) + 4x^2/(1-2x) \\ &= 4x + 2xa(x) + 4x^2/(1-2x) = 4x + 2xa(x) + (4x^2 - 1 + 1)/(1-2x) \end{aligned}$$

$$
\begin{aligned}
&= 4x + 2xa(x) - 1 - 2x + 1/(1-2x) = 2xa(x) - (1-2x) + 1/(1-2x), \text{ i.e.,} \\
(1-2x)a(x) &= -(1-2x) + 1/(1-2x), \text{ i.e.,} \\
a(x) &= -1 + 1/(1-2x)^2 \\
&= -1 + (1 + 2(2x) + 3(2x)^2 + \cdots + (n+1)(2x)^n + \cdots) \\
&= 0 + 2 \times 2x + 3 \times 2^2x^2 + \cdots + (n+1) \times 2^n x^n + \cdots .
\end{aligned}
$$

It follows that $a_n = \begin{cases} 0 & \text{if } n = 0, \\ (n+1)2^n & \text{if } n \geqslant 1. \end{cases}$

Method 4: Using generating functions on the original recurrence relation.

Let $a(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n + \cdots$. Using the original recurrence relation for a_n, $n \geqslant 1$, yields

$$
\begin{aligned}
a(x) &= a_0 + (a_0 + 2^2)x + (a_1 + a_0 + 2^3)x^2 + \cdots + \\
&\quad (a_{n-1} + a_{n-2} + \cdots + a_0 + 2^{n+1})x^n + \cdots \\
&= a_0 + x[(a_0) + (a_1 + a_0)x + \cdots + (a_{n-1} + a_{n-2} + \cdots + a_0)x^{n-1} + \cdots] + \\
&\quad 4x[1 + (2x) + (2x)^2 + \cdots + (2x)^{n-1} + \cdots] \\
&= a_0 + x(a_0 + a_1x + a_2x^2 + \cdots + a_nx^n + \cdots)(1 + x + x^2 + \cdots + x^n + \cdots) + \\
&\quad 4x/(1-2x) \\
&= a_0 + xa(x)/(1-x) + 4x/(1-2x), \text{ i.e.,} \\
\left(1 - \frac{x}{1-x}\right) a(x) &= a_0 + 4x/(1-2x) = 4x/(1-2x), \text{ i.e.,} \\
\left(\frac{1-2x}{1-x}\right) a(x) &= 4x/(1-2x), \text{ i.e.,} \\
a(x) &= \frac{4x(1-x)}{(1-2x)^2} = \frac{1-(1-2x)^2}{(1-2x)^2} = -1 + 1/(1-2x)^2 \\
&= -1 + (1 + 2(2x) + 3(2x)^2 + \cdots + (n+1)(2x)^n + \cdots) \\
&= 0 + 2 \times 2x + 3 \times 2^2x^2 + \cdots + (n+1) \times 2^n x^n + \cdots .
\end{aligned}
$$

Consequently, $a_n = \begin{cases} 0 & \text{if } n = 0, \\ (n+1)2^n & \text{if } n \geqslant 1. \end{cases}$