Algorithms (Fall 2008) Qual Exam

Answer ALL questions

Question 1 (25 points)

Show how to implement a queue using two stacks so that the amortized cost of each ENQUEUE and each DEQUEUE operation is $O(1)$.

Question 2 (15 points + 15 points + 5 points)

Consider the following recursive function (written in C style):

```
void f( int k ) {
   if (k > 0) f( rand(0,k-1) );
}
```

The function `rand(i,j)` returns a random integer from $\{i, i+1, i+2, \ldots, j\}$ with equal probability. Note that it is assumed that $i \leq j$.

Let a_n denote the average number of calls to `rand()` in computing $f(n)$. Note that $a_0 = 0$ and $a_1 = 1$.

(a) Give a recurrence for a_n for $n > 1$.

(b) Show that $a_n = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n} = H_n$, the n-th Harmonic number.

(c) Express[1] a_n in terms of n using Θ notation.

[1] Do not give $\Theta(H_n)$ as answer. You are expected to express the answer in terms of a common mathematical function.

Question 3 (30 points + 10 points)

Suppose there is a gambler with supernatural power who can guess correctly the outcome of each coin throw with the probability p, where $p > 1/2$. Suppose the bet amount is limited to an integer, and the amount won equals the amount staked.

Example: Let the probability of winning be 2/3. Suppose the gambler initially has 4 dollars. The gambler intends to make 3 bets, and would like to maximize his chance of earning 3 dollars or more. That is, together with the initial 4 dollars, he would want to have 7 or more dollars when he leaves the table. After some calculations, he comes up with the following strategy that maximizes his chance of achieving the goal:

```
bet 2 dollars                  // out of    4 dollars
if winning
   then bet 1 dollar           // out of    6 dollars
        if winning
           then stop playing   // has       7 dollars; prob = 4/9
           else bet 2 dollars  // out of    5 dollars
                if winning
                   then stop   // has       7 dollars; prob = 4/27
                   else stop   // ends with 3 dollars
   else bet 2 dollars          // out of    2 dollars
        if winning
           then bet 4 dollars  // out of    4 dollars
                if winning
                   then stop   // has       8 dollars; prob = 4/27
                   else stop   // ends with 0 dollars
           else stop           // ends with 0 dollars
```

The probability of achieving the goal of 7 or more dollars is computed to be 20/27. (Note that the last 4-dollar bet in the scheme can be changed to a 3-dollar bet without changing the probability of achieving the goal.)

Let `prob`(n, m, k) denotes the probability of achieving at least k total dollars with the **best** strategy for betting n times or less where m is the initial amount of money that the gambler has; that is, the gambler has won $k - m$ dollars or more.

(a) Give a recurrence, with the base cases, for computing `prob`(n, m, k).
(b) How much time, in Θ notation, does it take to compute `prob`(n, m, k) using dynamic programming?

Answers:

1. The top of stack 1 holds the tail item of the queue and the top of stack 2 holds the front item of the queue. We insert into stack 1 when we perform ENQUEUE and pop from stack 2 for DEQUEUE. When stack 2 is empty, then we move all elements from stack 1 to stack 2. Every ENQUEUE is charged two units of cost. One unit is for the actual work. Another unit is deposited together with the element. The deposit is needed to cover the cost when the element is moved from stack 1 to stack 2. Every DEQUEUE is charged one unit of cost. The copying of items from stack 1 to stack 2 are paid for by the deposit.

2. (a) $a_n = 1 + \frac{1}{n}\left(\sum_{i=0}^{n-1} a_i\right)$, or $a_n = 1 + \frac{1}{n}\left(\sum_{i=1}^{n-1} a_i\right)$ as $a_0 = 0$

2. (b) Base case: $a_1 = 1 = H_1$.

Induction Step: Rewriting the recurrence gives $\sum_{i=1}^{n-1} a_i = n(a_n - 1)$.

$$\begin{aligned}
a_n &= 1 + \tfrac{1}{n}\left(\textstyle\sum_{i=1}^{n-1} a_i\right) \\
&= 1 + \tfrac{1}{n}\left(\textstyle\sum_{i=1}^{n-2} a_i\right) + \tfrac{1}{n}a_{n-1} \\
&= 1 + \tfrac{1}{n}((n-1)(a_{n-1}-1)) + \tfrac{1}{n}a_{n-1} \\
&= a_{n-1} - \tfrac{n-1}{n} + 1 \\
&= a_{n-1} + \tfrac{1}{n} \\
&= H_{n-1} + \tfrac{1}{n} && \text{by the induction hypothesis} \\
&= H_n
\end{aligned}$$

2. (c) $a_n = H_n \approx \ln n = \Theta(\lg n)$

3. (a)

$\texttt{prob}(0, m, k) = 1$ if $m \geq k$ (or, $\texttt{prob}(n, m, k) = 1$ if $m \geq k$)

$\texttt{prob}(0, m, k) = 0$ if $m < k$

$$\texttt{prob}(n, m, k) = \max_{0 \leq i \leq m} p \cdot \texttt{prob}(n-1, m+i, k) + (1-p) \cdot \texttt{prob}(n-1, m-i, k)$$

3. (b) Each calculation of $\texttt{prob}(n, m, k)$ takes $O(m)$ time. For each n value, we need to consider different m values that range from 0 to k. The total time is $n \sum_{m=1}^{k} O(m) = n\ O(k^2) = O(nk^2)$.