CMPUT 204 - Seminar Practice Questions #1

Weeks of: September 9, 16 and 23, 2013

Scope. Problems involving mathematical equalities, relations, and summations that arise frequently in the counting arguments underlying the analysis of algorithms; inductive proofs and their applications; design and analysis of non-recursive algorithms, O-notation warm-up

Problems selected from the following list will be discussed in the seminars, as time permits.

1. [Summation]

   (a) Derive a closed form expression for $\sum_{i=a}^{n} i$, where a is an integer with $1 \le a \le n$. Hint: $\sum_{i=1}^{n} i = n(n+1)/2$

   (b) (The Telescoping method) Show that for any sequence $a_0, a_1, \cdots, a_n$ of numbers, $\sum_{k=1}^{n}(a_k - a_{k-1}) = a_n - a_0$ and $\sum_{k=0}^{n-1}(a_k - a_{k+1}) = a_0 - a_n$.

   (c) Using b) derive a closed form expression for $\sum_{k=1}^{n-1} \frac{1}{k(k+1)}$. Hint: define a_k such that $a_k - a_{k+1} = 1/(k(k+1))$.

2. [Logarithms] For $c > 0$, $\log_c x$ is defined by $c^{log_c x} = x$.

   (a) Using the basic definition of logarithms and no other property, prove the identity:

   $$\log_a x = \frac{\log_b x}{\log_b a},$$

   where a, b, and $x > 0$. This shows that logarithms to different bases only differ in constant factors.

   (b) What is $\log n^k$? Hint: $(a^b)^c = a^{bc}$

   (c) Using $\log(a \cdot b) = \log a + \log b$, part (b), and Stirling's approximation $n! \approx \sqrt{2\pi n}(n/e)^n$ give an approximation for $\log_e(n!)$.

3. [Mathematical Induction] Prove the following statements by mathematical induction:

   (a) $\sum_{i=1}^{n} i^2 \le n^3$ for all $n \ge 1$.

   (b) $\sum_{i=1}^{n}(2i-1) = n^2$ for all $n \ge 1$.

4. [Program Correctness]

   Prove that the following pseudo-code is correct, i.e., it returns $\max_{i=0..n-1} A[i]$.

```
// assume n >= 1
// input: array of n numbers
// output: maximum value in the array
function max(A[0..n-1])
m = A[0]
i = 1
while i < n do
  if A[i] > m then
    m = A[i]
  end
```

```
  i = i + 1  (*)
end
return m
```

Hint: First show, that the program always terminates. Then prove that the loop invariant ($m = \max_{j=0..i-1} A[j]) \;\wedge\; i \leq n$ holds before the program enters the while-loop and right after line (*) when the loop body was executed. Finally, prove that the loop invariant and the loop exit condition together imply the program's correctness.

5. [Pseudo Code] You have available the following global variables: a constant N, an array $Q[0 \cdots N]$ of $N+1$ integers, and two variables $Qfront$ and $Qrear$. Without using new variables (not even a temporary variable!), explain how to implement a *queue* (first-in first-out) data structure that can hold at most N integers and executes each of the following basic operations in constant time (i.e. the time is independent of N):

   (a) `Qinit(Q)`: initialize Q to be empty.
   (b) `Isempty(Q)`: return 1 if Q is empty, else return 0.
   (c) `Isfull(Q)`: return 1 if Q is full, else return 0.
   (d) `Qadd(x,Q)`: if Q is not full then add x to the end.
   (e) `Qremove(Q)`: if Q is not empty then remove the item at the front.

   Write pseudo-code for each of the above functions.

6. [Induction and Recursive Algorithms] Use induction to prove that the following recursive algorithm computes $3^n - 2^n$ for all $n \geq 0$.

```
// assume n >= 0
function g(n)
if n <= 1 then
    return n
end
return 5*g(n-1) - 6*g(n-2)
```

7. [O-notation]

   (a) Prove or disprove: $2^{n+1} \in O(2^n)$, $2^{2n} \in O(2^n)$
   (b) Show that $O(1)$ is the set of bounded functions.
   (c) Explain why the statement "The runtime of algorithm A is at least $O(n^2)$" is meaningless.

The following summations frequently arise when analyzing algorithms. You may use them in solving problems.

$\sum_{i=2}^{n} \frac{1}{i} \leq \ln n$	$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$	$\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$	$\sum_{i=1}^{n} i2^i = (n-1)2^{n+1} + 2$
	$\sum_{i=1}^{n} i^2 = \frac{2n^3+3n^2+n}{6}$	$\sum_{i=0}^{n} \frac{1}{2^i} = 2 - \frac{1}{2^n}$	
