CSci 1302 Assignment 8

Due Friday, April 2nd, 2004 (Problems 5-7 due Monday, April 5th)

Part I: Induction.

Problem 1 (15 points). Prove the following by induction. Make sure to read the claims carefully. It might be helpful to write out and check the first few cases before proving the general formula.

1. $\sum_{i=0}^{n}(2 \cdot i + 1) = (n+1)^2$ (note that the summation starts at $i = 0$, not $i = 1$).
2. $\sum_{i=1}^{n}(i \cdot 2^i) = (n-1) \cdot 2^{n+1} + 2$.
3. $4^n - 1$ is divisible by 3 for $n \geq 1$.

Problem 2 (5 points). Prove the following property of Fibonacci numbers by induction on n (m remains constant) $F_{n+m} = F_m \cdot F_{n+1} + F_{m-1} \cdot F_n$, assuming that $n \geq 0$ and $m \geq 1$.

Problem 3 (Extra credit, 5 points). Consider a currency consisting of 2-cent and 5-cent coins only. Show by induction that any amount above 3 cents can be represented using these coins.

Hint: suppose you are given a bag of coins whose sum is n ($n > 3$). How can you replace some coins in the bag to increment the total amount in the bag by 1?

Part II: Algorithm analysis.

Problem 4 (6 points). Prove the following statements:

1. $n^3 + 1 \in O(n^3 - 1)$
2. $n^3 - 1 \in O(n^3 + 1)$
3. $20n^2 + 100 \in O(n^3 + 1)$

Problem 5 (6 points) Consider the following algorithm for reversing the order of elements in an array (written in a Java-like syntax):

```
// Given: an array a[n] of n integer elements
// Result: the elements of the array are reversed.
// The index of the first element is 1

int i = 1;
int j = n;
while (i < j) {
   // switching the i-th and the j-th elements
```

```
   int temp = a[i];
   a[i] = a[j];
   a[j] = temp;
   i = i + 1;
   j = j - 1;
}
```

Show how the algorithm works on the array `[1, 2, 3, 4]`. Compute the run-time of the algorithm based on the number of assignment statements (note that there are 3 assignments in each run of the loop). Give the "Big-O" approximation for the run-time (i.e. $O(n), O(n^2)$, etc.). Show your computations.

Problem 6 (6 points) Consider another algorithm for reversing elements of an array. This algorithm is recursive, so we write it as a recursive function. You may assume that the number of elements in the array is a power of 2. We use the notation `a[i..j]` to denote a portion of the array from index i to index j.

The idea of the algorithm is to divide the array in half, reverse each sub-array recursively, and then switch the two halves.

```
// Given: an array a[n] of n integer elements
// Result: the elements of the array are reversed.
// The index of the first element is 1

void reverse (a[1..n]) {
   if (n < = 1) return; // do nothing
   else {
        reverse (a[1..n/2]);
        reverse (a[((n/2) + 1)..n]);
        // switch the two halves:
        int i = 1;
        while (i <= n/2) {
           int temp = a[i];
           a[i] = a[(n/2) + i];
           a[(n/2) + i] = temp;  // corrected 4/2/04
           i = i + 1;
        }
   }
}
```

Show how the algorithm works on the array `[1, 2, 3, 4]`. What is the running time of the algorithm? What is the "Big-O" approximation for the run-time? Show your computations.

Problem 7 (2 points) Which of the algorithms in the previous two problems would you recommend for reversing an array? Explain your reasoning.