MA/CSSE 473 – Design and Analysis of Algorithms

Homework 3 (10 problems, 60 points total)

Turn this in to the HW 03 drop box on ANGEL. Complete the survey for extra credit.

When a problem is given by number, it is from the Levitin textbook. 1.1.2 means "problem 2 from section 1.1" Numbers in [square brackets] are problem or page numbers from Levitin's 2nd edition.

Problems for enlightenment/practice/review (not to turn in, but you should think about them):

How many of them you need to do serious work on depends on you and your background. I do not want to make everyone do one of them for the sake of the (possibly) few who need it. You can hopefully figure out which ones you need to do.

3.1.8 [3.1.5] (selection sort practice)
3.1.13 [3.1.10] (Is bubble sort stable?)

Problems to write up and turn in:

1. (5) Prove by mathematical induction that the following formula is true for every positive integer n.
$$\sum_{i=1}^{n}(-1)^{i+1}i^2 = \frac{(-1)^{n+1}n(n+1)}{2}$$
2. (8) Prove (not necessarily directly by mathematical induction) that $\sum_{i=1}^{n} i \cdot r^i < \frac{r}{(1-r)^2}$ for all n≥1 and 0 < r < 1.
This is a hard problem. Stat early. Use the formula for the sum of a geometric sequence.
3. (8) Let F_n be the n^{th} Fibonacci number (recall that $F_0 = 0$ and $F_1 = 1$ in our formulation). Show by mathematical induction that that for all n> 0,
$$\sum_{i=1}^{n} F_i^{\,2} = F_n F_{n+1}$$
4. (8) Prove by mathematical induction that F_n (as defined above) is even if and only if n is divisible by 3.
Be sure to show both directions of the "if and only if".
5. (4) 3.1.2 [3.1.2] (algorithms for computing a^n)
6. (7) 3.1.4 [3.1.4] (polynomial evaluation)
7. (2) 3.1.9 [3.1.6] (stability of selection sort)
8. (2) 3.1.10 [3.1.7] (selection sort linked list)
9. (8) 3.1.14 [3.1.11] (alternating disks) Come up with the best solution that you can, and come up with a formula for the number of moves as a function of N, the total number of disks. Show how you get your formula.
You may assume that N is even.

10 (8) Here is an inefficient algorithm for computing F(N):

```
F(n):
        if n ≤ 1 return n
        else return F(n-1) + F(n-2)
```

Once again I ask you to prove by induction what Levitin derived by other means. The idea of this problem comes from Weiss Section 7.3. Use mathematical induction to show that the formula (in the last line of the Weiss excerpt below) is indeed the solution to the given recurrence (the one in the next-to-last line of the Weiss excerpt).

Let $C(N)$ be the number of calls to `fib` made during the evaluation of `fib(n)`. Clearly $C(0) = C(1) = 1$ call. For $N \geq 2$, we call `fib(n)`, plus all the calls needed to evaluate `fib(n-1)` and `fib(n-2)` recursively and independently. Thus $C(N) = C(N-1) + C(N-2) + 1$. By induction, we can easily verify that for $N \geq 3$ the solution to this recurrence is $C(N) = F_{N+2} + F_{N-1} - 1$. Thus the number