MA/CSSE 473 – Design and Analysis of Algorithms

Homework 3 (11 problems, 72 points total) Updated for Summer, 2017

Turn this in to the HW 03 drop box on Moodle. Optional: complete the HW3 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.

Many of these problems review concepts/techniques from CSSE 230.

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. hopefully you can 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) 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,
$$\sum_{i=1}^{n}F_i^{\,2} = F_nF_{n+1}$$

3. (8) Prove by mathematical induction that F_n (as defined at the beginning of the previous problem) is even if and only if n is divisible by 3. Be sure to show both directions of the "if and only if".
Suggestion: prove both directions in one induction proof.

4. (4) 3.1.2 [3.1.2] (algorithms for computing a^n) For this problem, assume that a single numeric multiplication can be done in constant time.

5. (7) 3.1.4 [3.1.4] (polynomial evaluation) Again assume that a single numeric multiplication is constant time.

6. (10) 3.1.5 [not in 2nd edition, see HW 3 problems document] (points: ring 4, star 3, mesh 3).
Note that the ring, star, and mesh are three separate algorithms, and that it is possible that a graph is none of these three.

   Also note that there are no loops (edges from a vertex to itself) in graphs in our textbook unless a specific problem says that they are allowed. That is stated at the top of p. 29 [middle of p. 29].

   Previous note from Piazza:

   First, every matrix that actually represents an undirected graph *is* symmetric, so you do not have to test for that. Next, because it is symmetric, in many cases it is only necessary to test rows, not rows and columns. Notice the **if any** part of the problem description; "none of the above" is a possible answer. The ring case is the trickiest. Several students have come to me with a simple test that is *too* simple, in the sense that every ring passes their test, but some non-rings also pass their test.

7. (2) 3.1.9 [3.1.6] (stability of selection sort)

8. (2) 3.1.10 [3.1.7] (selection sort linked list)

(8) 3.1.14 [3.1.11] (alternating disks) Come up with the best solution that you can, and come up with a formula (exact, not just big-O) for the number of moves as a function of N. N is the number of disks of each color; i.e. if 8 black disks and 8 white ones, N=8. Show how you get your formula.

Previous note from Piazza:

> If your algorithm ever asks what color any specific disk is, then that algorithm is probably not optimal. We know in advance where all of the white disks are and where all of the black disks are, so why should we ever have to ask. An algorithm whose number of swaps and/or number of comparisons is the same as Bubble Sort will not receive much credit. You can do better than that!

9. (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 something that 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 recurrence in the next-to-last line of the following Weiss excerpt).

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 $C(N) = F_{N+2} + F_{N-1} - 1$.

10. (10) King in N moves. In chess, a king can move one space in any of the 8 directions (2 vertical, 2 horizontal, 4 diagonals). Suppose our chessboard is infinite in every direction, and a king starts on a particular square.
    (a) How many different squares are possible locations for the king after N or fewer moves?
    (b) Answer the same question for a “crippled king” that can only move horizontally and vertically (no diagonal moves allowed).

    For both parts, show how you get your answer.

Previous questions and answers from Piazza:

Q: Is the king's position inside a square, or is it one of the corners of the square? **A:** Inside the square.
Note: Be sure to notice "or fewer" in the problem statement. It makes the problem easier.
Note: "Show how you get your answer" does not require a formal induction proof. A diagram will probably suffice.