Recitation 12

- Here is a set of additional problems. They range from being very easy to very tough. The best way to learn the material in 310 is to solve problems on your own.
- Feel free to ask (and answer) questions about this problem set on Piazza.
- This is an **optional** problem set; do not turn this in for grading.
- While you don't have to turn this in, be warned that this material **can** appear in a quiz or exam.

---

1. Suppose that $F_0 = 0, F_1 = 1$ and $F_n = F_{n-1}+F_{n-2}$. Prove using induction that for every $n \geq 1$,

$$F_1^2 + F_2^2 + \dots F_n^2 = F_n \cdot F_{n+1}.$$

2. Consider the following algorithm in pseudocode form:

function $\textsc{sq}(n)$:

  $S \leftarrow 0, i \leftarrow 0$

  while $i < n$

    $S \leftarrow S + n$

    $i \leftarrow i + 1$

  return S

Two questions:

(i) Figure out what the algorithm is supposed to be doing, and using induction, prove that the algorithm is correct.

(ii) Analyze the algorithm's efficiency (in terms of running time), assuming that each add operation takes a unit amount.

3. Consider the following "algorithm" that prints out a bunch of things (it is unimportant what it prints out):

for $(i = n; i \geq 1; i = i/2)$:

  for j in $[1, i]$:

    print xyz

Assume that each print command takes 1 unit of time. Using big-Oh notation, estimate the running time of the algorithm as a function of n.

4. Two algorithms (call them A and B) have different time complexities. The first algorithm exhibits time complexity $T_A(n) = 5n \log_{10} n$ microseconds, and the second exhibits time complexity $T_B(n) = 25n$ microseconds, for a problem of input size n. Which algorithm is better in the Big-Oh sense? For which problem sizes does it outperform the other?