15-451/651 Algorithms, Spring 2019
Recitation #1 Worksheet

-IV. Asymptotic analysis. [1] For each pair $\langle f, g \rangle$ of functions below, list which of the following are true: $f(n) = o(g(n))$, $f(n) = \Theta(g(n))$, or $g(n) = o(f(n))$.

1. $f(n) = \ln(n), g(n) = \log_{10}(n)$.

2. $f(n) = n^{1.5}, g(n) = n \log^4(n)$.

3. $f(n) = 2^{2n}, g(n) = 2^n$.

4. $f(n) = n^{0.001}, g(n) = (\log_2 n)^{100}$.

-III. Solving recurrences by unrolling. Solve the following recurrences in Θ notation by unrolling (assume all base cases have $T(1) = 0$):

1. $T(n) = 1 + 2T(n-1)$.

2. $T(n) = n + T(n/2)$.

3. $T(n) = n + 2T(n/2)$.

[1] The Romans did not have negative numbers, so we're playing fast and loose with notation here.

-II. Solving by guess and inductive proof. Consider the recurrence $T(n) = 2T(n/2) + \log_2(n)$ with base case $T(1) = 0$. This is sometimes called the "heapsort recurrence" and can be viewed as the sum of the heights of all nodes in a completely balanced binary tree of n leaves, where leaves are at height 0, their parents are at height 1, etc.

(a) Show that $T(n) = O(n \log n)$ and $T(n) = \Omega(n)$.

But what's the truth? One way to solve it is to try small cases ($n = 1, 2, 4, 8, 16, 32$) and notice that this seems to fit the pattern $T(n) = 2n - \lg(n) - 2$.

(b) Prove by induction that $T(n) = 2n - \lg(n) - 2$ indeed is the solution.

(c) Can you think of other ways to prove $T(n) = \Theta(n)$? (E.g., you could unroll the recursion, or give a "combinatorial proof".)

-I. Using the master formula. Solve the following recurrences (usual base case) using the master formula.

1. $T(n) = 3T(n/2) + n^2$.

2. $T(n) = 4T(n/2) + n^2$.

3. $T(n) = 5T(n/2) + n^2$.

I. Probability Facts. Let's follow the convention of using uppercase letters X, Y, Z to denote *random variables* (which we often abbreviate to *r.v.*s).

Independence: Random variables X, Y are *independent* if for any values a, b, we have

$$\mathbf{Pr}[X = a, Y = b] = \mathbf{Pr}[X = a] \cdot \mathbf{Pr}[Y = b].$$

1. Consider flipping two fair coins. Let $X \in \{0, 1\}$ be the outcome of the first coin (where we think of heads as 1 and tails as 0) and let $Y \in \{0, 1\}$ be the outcome of the second coin. Let $Z = X + Y \bmod 2$. Are X and Y independent? What about X and Z?

Linearity of expectation: Given any two r.v.s X, Y,

$$\mathbf{E}[X + Y] = \mathbf{E}[X] + \mathbf{E}[Y].$$

This is true even if X and Y are not independent!

2. Suppose we take n books numbered 1 to n, and place them on a shelf in a (uniformly) *random* order. What is the expected number of books that end up in their correct location? ("correct" = location it would be in if they were sorted).

Markov's inequality: given a *non-negative random variable* X with mean/expectation $\mu = \mathbf{E}[X]$,

$$\mathbf{Pr}[X > c\,\mu] \leq \frac{1}{c}.$$

(Exercise: give a proof!)

3. Show that if X is allowed to take negative values then the above inequality is no longer true.

II. Sorting by Swaps. Imagine a sorting algorithm that somehow picks two elements that are out of order with respect to each other (not necessarily adjacent, as in `insertion-sort`) and swaps them. *Question: can we argue that such a procedure (no matter how stupidly it picks the elements to swap so long as they were out of order) has to terminate (i.e., the number of swaps it will perform is bounded)?*

To do this, one good way is to find a potential function: a finite, non-negative quantity that is strictly reduced with each swap. Any ideas? One quantity that works is the total number of pairs of elements that are out of order w.r.t. each other (this is called the number of *inversions*). When a swap is performed, clearly the inversion of those two is removed, but notice that new inversions may be created. (e.g., in $[5, 1, 2]$, swapping 5 and 2 creates a new inversion between 1 and 2).

4. Show the total number of inversions goes down with each swap.

III. Breaking Eggs. Say I choose a number between 1 and N. You want to guess it in as few questions as possible (each time you make an incorrect guess, I'll tell you if it is too high or too low). As we know, the strategy that minimizes the worst-case number of guesses is to do binary search: it takes $\lceil \log_2 N \rceil$ guesses.

But, what if you are only allowed **one** guess that is too high? Can you still solve the problem in $o(N)$ guesses? [If you were not allowed **any** guesses that are too high, the only option you would have would be to guess 1,2,3,... in order].

Q: Show how to figure out how many centimeters high you can drop an egg without it breaking, using $2\sqrt{N}$ guesses, when you only have two eggs. (For answer, see footnote[2])

[2]Here's one strategy: guess $\sqrt{N}, 2\sqrt{N}, \cdots$ until you get to some $(i+1) \cdot \sqrt{N}$ that's too high. Now the number is between $i\sqrt{N}$ and $(i+1)\sqrt{N}$ so we can finish it off in $\sqrt{N}$ more guesses. #Guesses $\leq 2\sqrt{N}$.

Q: Can you get down to $\sqrt{2N} \approx 1.42\sqrt{N}$ guesses?

Q: Can you show an $\Omega(\sqrt{N})$ lower bound for any deterministic algorithm? (Hint.[3])

Q: Show upper/lower bounds of $\Theta(n^{1/3})$ if you're allowed *two* guesses that are too high?

[3]Hint: What if the algorithm makes a guess g_i that is at least $\sqrt{N}$ larger than any previous guess? And what if the algorithm *never* makes such a guess?