15-451 Algorithms, Spring 2016
Recitation #2 Worksheet

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?

   Solution: X and Y are independent by definition. X and Z are also independent, despite us defining Z in terms of X. To prove this, just calculate $\mathbf{Pr}[X = a, Z = c]$ for each setting of a, c and show that it equals $\mathbf{Pr}[X = a] \cdot \mathbf{Pr}[Z = c]$.

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

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

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).

   Solution: Let X_i be the "indicator" r.v. which is 1 when the i^{th} book is in the correct location, and 0 otherwise. Note that $\mathbf{E}[X_i] = \mathbf{Pr}[X_i = 1] =$ the probability that the i^{th} book is in the correct location, which is $1/n$. Now $X = \sum_{i=1}^{n} X_i$ is the number of books in the correct location, so $\mathbf{E}[X] = \mathbf{E}[\sum_i X_i] = \sum_i \mathbf{E}[X_i] = \sum_i (1/n) = n \cdot 1/n = 1$.

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.

   Solution: Suppose $X = 1$ with probability 0.51 and -1 with probability 0.49. Then $\mu = \mathbf{E}[X] = 0.02$, but $\mathbf{Pr}[X > 5\mu] = \mathbf{Pr}[X > 0.1] = 1/2$ which is more than $1/5$.

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. *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.

   Solution: Suppose we swap the elements (say a and b) at locations i and j. Then we have $a > b$. This removes the inversion between a and b. If we have a new inversion between b and c, for some element c at position k lying between i and j, then there since $a > b$ and $b > c$, there was a previous inversion between a and c which has disappared. Similarly, any new inversions between c and a mean there earlier c and b also gave an inversion. Hence every inversion in the new ordering corresponds to one in the old ordering, and one inversion (a, b) has disappeared.

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 is to guess 1,2,3,... in order].

Q: Any ideas? [To restate the problem: you want to figure out how many centimeters high you can drop an egg without it breaking, and you only have two eggs.] (Spoiler below.[1])

Solution: See the hint.

Q: Can improve the constant to below 2?

Solution: Consider the strategy that queries the position k, then while you get "too low", query the positions $k + (k-1), k + (k-1) + (k-2), \ldots$ all the way up to $\sum_{i=0}^{k-1}(k-i) = \frac{k(k+1)}{2}$. And once you get "too high" at any position, do linear search from the previous "too low" location. Observe: the differences decrease by 1, so you will never make more than k queries. To make sure we cover all locations from 1 to N, we want k large enough such that $\frac{k(k+1)}{2} \geq N$. Setting $k = \lceil \sqrt{2N} \rceil$ ensures this.

[1] Here's one strategy: guess $1, \sqrt{N}, 2\sqrt{N}, \cdots$ 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. Total number of guesses is at most $2\sqrt{N}$.

Q: Can you show an $\Omega(\sqrt{N})$ lower bound for any deterministic algorithm? (Hint below.[2])

Solution: Here's a strategy for the adversary. Let $p_0 = 0$. Let p_1 be the first location the algorithm queries, p_2 where it queries if it gets a "too low", p_3 where it queries if it gets a "too low" at p_2, etc. We may assume that $p_i > p_{i-1}$. Suppose each $p_i \leq p_{i-1}+\sqrt{N}$, then the adversary can keep answering "too low" and make the algorithm incur cost at most $\lfloor \frac{N}{\sqrt{N}} \rfloor = \Omega(\sqrt{n})$.

If not, if the algorithm has some i such that $p_i > p_{i-1} + \sqrt{N}$, then the adversary says "too high" at p_i. Now the algorithm is forced to do linear search between p_{i-1} and p_i; the adversary can say "too low" at the locations $p_{i-1} + 1, p_{i-1} + 2, \ldots, p_i - 2$ to incur cost $\Omega(\sqrt{N})$ again.

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

Solution: First do $n^{2/3}$ increments, and then $n^{1/3}$ within that, and then linear queries within that. The outermost loop will do at most $n/n^{2/3} = n^{1/3}$ queries. And then the middle one will do at most $n^{2/3}/n^{1/3} = n^{1/3}$ queries. And the inner one will also do at most $n^{1/3}$ using linear search. So a total of $3 \times n^{1/3}$. In general, you can do $k \times n^{1/k}$ for k eggs. This is not tight, can you do a little better?

[2]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?