Read Chapters 1.0, 1.4, 2, 5.6 from the textbook, and optionally Chapters 6.2 and 6.3 from the book by Mitzenmacher/Upfal. Here are the homework problems:

(1) In class, we analyzed the Randomized Quicksort algorithm. Another way to avoid choosing too many bad pivot elements in Quicksort is to uniformly permute the input array, and then just run standard Quicksort (say, using the last element of a (sub-)array as a pivot each time). Intuitively, it feels like this should also give us a “uniformly random pivot” each time, and therefore make the algorithms basically the same. Prove this intuition by showing the following: for each number $c$, we have that
\[
\text{Prob}[\text{Randomized Quicksort uses exactly } c \text{ comparisons}] = \text{Prob}[\text{Quicksort with uniformly random shuffling uses exactly } c \text{ comparisons}].
\]

Hint: This should seem intuitively very obvious, so the interesting question is how to make the intuition precise. You may want to somehow couple the executions of the two algorithms. In doing so, a very useful technique is the “Principle of Deferred Decisions” (which is often useful in analyzing random processes). Basically, it states that you can leave random decisions until the point in time when they are needed, or make them earlier, as you please, without changing the overall distribution. You can read more in Section 3.5 of the textbook (which we won’t explicitly cover in class).

As an example, suppose that you have an algorithm that draws a number $x$ uniformly randomly from $[0, 1]$. Based on whether $x < \frac{1}{3}$ or $x \geq \frac{1}{3}$, it makes some decisions, and then does further computation involving $x$. Then, you can equivalently state the algorithm by not drawing $x$ at the beginning. At the moment when the algorithm tests for $x < \frac{1}{3}$, you decide with probability $\frac{1}{3}$ that it is, and with probability $\frac{2}{3}$ that it isn’t. Later on, when the actual value of $x$ is needed, in the first branch, you generate it uniformly randomly from $[0, \frac{1}{3})$, and in the second branch uniformly randomly from $[\frac{1}{3}, 1]$. The Principle of Deferred Decisions then states that the outcomes of the two algorithms are the same.

(2) Use the method of conditional expectations to derive a deterministic, polynomial-time, $7/8$-approximation algorithm for MAX-3-SAT, and prove its approximation guarantee. Assume that each clause contains exactly 3 literals. Notice that your algorithm should be entirely deterministic, i.e., not make any references to “expectation”, “probability”, or such (though of course, your proof can use these concepts).

Hint: what you get out may not be the “obvious” algorithm you were expecting at the beginning. If it is, double-check your analysis.

(3) You want to evaluate the integral of functions $f : [0, 1] \rightarrow \mathbb{R}$, but you want to do so super-efficiently. So you have come up with the following algorithm: Choose a point $x \in [0, 1]$, and evaluate and output $f(x)$. This is clearly not always correct, so we want to know how far from the correct answer it is in the worst case. If the function $f$ could jump around arbitrarily, you would have no chance of ensuring anything, so we will assume that for all $x, y \in [0, 1]$, you have $|f(x) - f(y)| \leq |x - y|$.

The correct answer is $\int_0^1 f(t)dt$, whereas you output $f(x)$ for one value $x$. Thus, your absolute error is $|f(x) - \int_0^1 f(t)dt|$; if you choose $x$ from a distribution, it is $\mathbb{E} \left[|f(x) - \int_0^1 f(t)dt|\right]$. Prove the following:

(a) For every deterministic choice of $x$, there is an input function $f$ such that the absolute error is at least $\frac{1}{4}$.

(b) If you choose $x \in \{\frac{1}{3}, \frac{2}{3}\}$ with probability $\frac{1}{2}$ each, the expected absolute error is at most $1/6$.

(c) For every distribution over $x \in [0, 1]$, there is an input function $f$ such that the absolute error is at least $1/8$. 

1
If you are really ambitious, you can try to find a distribution and lower bound proof that actually match. (Hint: the value at which they match is $1 - \sqrt{3}/2$.) This is significantly more difficult, and hence not part of the assignment.

(4) Let $G$ be a (directed) graph, $s$ a source, and $t$ a sink. Two players play the following “intrusion” game. Player 1 picks a path $P$ from $s$ to $t$ (possibly randomly), while player 2 picks a set $S$ of $r$ edges of $G$. Think of player 1 taking $P$ to get from $s$ to $t$, while player 2 places checkpoints (or patrols) on the edges in $S$. If $P \cap S = \emptyset$, then player 1 manages to get to $t$ undetected, and wins. If $P \cap S \neq \emptyset$, then player 2 catches player 1, and wins. Player 1 wants to minimize the probability of being caught, while player 2 wants to maximize the probability of catching player 1. Let $M$ be the number of edges in a minimum $s$-$t$ cut.

(a) Give a (possibly randomized) strategy for player 1 to be caught with probability at most $\min(r/M, 1)$.

(b) Prove that no strategy (randomized or deterministic) can have a probability of being caught strictly less than $\min(r/M, 1)$ against a player 2 who plays perfectly.

**Open Problem:** Suppose that there are two sinks $t_1, t_2$. If player 1 manages to get to $t_1$ undetected, he wins one point. If he gets to $t_2$ undetected, he wins two points. If he is caught, player 2 wins. (It doesn’t matter if we call this 0 or 1 points for player 2.) Give an algorithm that computes an optimal randomized strategy for player 1 and/or player 2 (assuming that that player must go first).