15-251: Great Theoretical Ideas In Computer Science

Recitation 12 : Randomized Algorithms

Lecture Review

- A *randomized algorithm* is an algorithm that has access to random bits, i.e. it can flip a coin. In this class we will allow randomized algorithms to call RandInt(n) and Bernoulli(p).
- Here are two interesting classes of randomized algorithms:
  - An algorithm A is a $T(n)$-time *Las Vegas* algorithm if
    - A always outputs the right answer, and
    - for every input $x \in \Sigma^*$, $\mathbf{E}$[number of steps $A(x)$ takes] $\leq T(|x|)$.
  - An algorithm A is a $T(n)$-time *Monte Carlo* algorithm with error probability ε if
    - for every input $x \in \Sigma^*$, $A(x)$ gives the wrong answer with probability at most ε, and
    - for every input $x \in \Sigma^*$, $A(x)$ has a worst-case running-time of at most $T(|x|)$.
- We can use *boosting* to improve the success probability of Monte Carlo algorithms via repeated trials.

A Hard Exam

(a) Suppose that the average score on the latest 15-150 exam was 10 points out of 100 and that 200 students took the exam. What's an upper bound on the number of students who received a perfect score? Assume that the 150 TAs are kind enough to not assign negative scores to students.

(b) **Markov's inequality**: Let $\boldsymbol{X}$ be a non-negative random variable with non-zero expectation. For any $c > 0$,

$$\Pr[\boldsymbol{X} \geq c\,\mathbf{E}[\boldsymbol{X}]] \leq \frac{1}{c}.$$

(No need to prove this — refer to the course notes to see the proof. But note that the proof is similar to the reasoning in part (a).)

What happens in Las Vegas doesn't stay in Monte Carlo

The expected number of comparisons that the Quicksort algorithm makes is at most $2n \ln n$ (which you can cite without proof — you might see a proof of this fact if you take 15-210). Describe how to convert this Las Vegas algorithm into a Monte Carlo algorithm with the worst-case number of comparisons being $1000n \ln n$. Give an upper bound on the error probability of the Monte Carlo algorithm.

Randomization Meets Approximation

3SAT is a hard problem to solve exactly, but is it hard to find a decent approximation algorithm for? (Maybe not!)

Consider the MAX-3SAT problem where, given a CNF formula in which every clause has exactly 3 literals (with distinct variables), we want to find a truth assignment to the variables in the formula so that we maximize the number of clauses that evaluate to True.

Describe a polynomial-time randomized algorithm with the property that, given a 3CNF formula with m clauses, it outputs a truth assignment to the variables such that the expected number of clauses that evaluate to True is $\frac{7}{8}m$ (i.e., in expectation, the algorithm is a $\frac{7}{8}$-approximation algorithm).

(Extra) (Brain Teaser) Passive-Aggressive Passengers

Consider a plane with n seats $s_1, s_2, \ldots, s_n$. There are n passengers, $p_1, p_2, \ldots, p_n$ and they are randomly assigned unique seat numbers. The passengers enter the plane one by one in the order $p_1, p_2, \ldots p_n$. The first passenger p_1 does not look at their assigned seat and instead picks a uniformly random seat to sit in. All the other passengers, p_2, p_3, ... , p_n, use the following strategy. If the seat assigned to them is available, they sit in that seat. Otherwise they pick a seat uniformly at random among the available seats, and they sit there. What is the probability that the last passenger, p_n, will end up sitting in their assigned seat?