CS 161 Fall 2017: Section 4/Midterm Review

Randomized Algorithms: Matrix Multiplication Verification

We have three matrices, A, B and C, each of size $n \times n$. We want to verify if $AB = C$. We could do so by multiplying and verifying them in $O(n^3)$ (or faster using some divide-and-conquer tricks!), but we want to develop a faster randomized algorithm here.

(a) Let $\mathbf{v}$ be a vector with n elements, each 0 or 1 with probability $\frac{1}{2}$. If we have another nonzero vector $\mathbf{u}$, show that $\mathbf{uv} = 0$ with probability $\leq \frac{1}{2}$. (Hint: express $\mathbf{uv}$ as $\mathbf{u}_k\mathbf{v}_k + x$ where $\mathbf{u}_k \neq 0$.)

(b) Note that the above extends to a matrix M: if M is nonzero, we have that $M\mathbf{v} = 0$ with probability $\leq \frac{1}{2}$. Given this, show that $P[AB\mathbf{v} = C\mathbf{v}] \leq \frac{1}{2}$

(c) Using the previous parts, develop a randomized algorithm to check if $AB = C$.

(d) Is your above algorithm a Las Vegas or Monte Carlo algorithm?

(e) What is the expected runtime of your algorithm? What is the probability of returning a correct answer?

Selection Potpurri

(a) You are implementing the SELECT, and your friend is writing the pivot selection portion using median of medians. However, it turns out that your partner left some bugs, so the only guarantee you have is that pivot selection will return some element of the array. Given that you implemented your portion of SELECT correctly, will your program still work? What is the worst-case runtime?

(b) Instead of using median of medians, you decide to use the mean μ of the array. (To guarantee an element in the array, assume that this is the smallest element in the array $\geq \mu$.) What is the worst-case runtime of SELECT?

(c) You have a set S of n integers, and you're given integer $1 \leq k < n$. Design an algorithm to verify that

$$\forall T \subset S, |T| = k, \sum_{t \in T} t \geq k$$

In other words, for every subset of S of size k, verify that the sum of elements in that subset is $\geq k$.

Number Distances

We are given an unsorted array A with n numbers between 1 and M, where M is a large (but constant) positive integer. We want to find if there exist two elements of the array that are within T of one another.

(a) Design a simple algorithm that solves this in $O(n^2)$.

(b) Design an algorithm that solves this in $O(n \log n)$.

(c) How could you solve this in $O(n)$? (Hint: modify bucket sort!)