Complexity for Probabilists
Comments and partial solutions to homework 3

This time Problem 1 seemed to be by far the most difficult for most of you, while Problems 2 and 3 seem to have caused less trouble.

1. (a) Run the algorithm A n times, let

$$X_i = \begin{cases} 1 & \text{if the } i^{\text{th}} \text{ run gives the right answer} \\ 0 & \text{otherwise} \end{cases}$$

and let $X = \sum_{i=1}^{n} X_i$. The X is binomially distributed with parameters n and p, so that $\mathbf{E}[X] = np$ and $\mathbf{Var}[X] = np(1-p)$ for some $p \geq \frac{2}{3}$. We only need to consider the worst case $p = \frac{2}{3}$, when $\mathbf{E}[X] = \frac{2n}{3}$ and $\mathbf{Var}[X] = \frac{2n}{9}$, and we get

$$\begin{aligned} \mathbf{Pr}(A^* \text{ gives the wrong answer}) &\leq \mathbf{Pr}\left(X \leq \frac{n}{2}\right) \\ &\leq \mathbf{Pr}\left(\left|X - \frac{2n}{3}\right| \geq \frac{n}{6}\right) \\ &\leq \frac{\mathbf{Var}[X]}{\left(\frac{n}{6}\right)^2} = \frac{8}{n}, \end{aligned}$$

where the last inequality is an application of Chebyshev's inequality. Taking $n \geq \frac{8}{\delta}$ gives

$$\mathbf{Pr}(A^* \text{ gives the wrong answer}) \leq \delta .$$

(b) Using either standard large deviations theory, or Håstad's elementary reasoning on p. 92 of his lecture notes, we get an estimate which grows like $\log(\frac{1}{\delta})$ rather than like $\frac{1}{\delta}$ when $\delta \to 0$, which is a huge improvement. The moral here is that although Chebyshev's inequality is a very widely applicable tool, it often yields answers that are far from sharp.

2. Let $p(n)$ be the maximum number of random bits used by the algorithm on an input of length n. The probability of getting only 0's as the random bits is $2^{-p(n)} > 0$, and since the algorithm is a Las Vegas algorithm, we know that "all 0's" yields the right answer. Hence we can replace the random bits by 0's, to get a polynomial time deterministic algorithm.

3. Most of you came up with the following very natural algorithm: Group the X_i's into pairs, ignore 00's and 11's, and interpret 10 as a 1 and 01 as a 0.

   For those of you who are *really* ambitious (and perhaps know some information theory, which is helpful), here's another nice problem to think about: Can you find algorithms which are more efficient, in the sense that the average number of X_i bits needed to produce each Y_j bit is smaller? What is the best possible efficiency? (The answer to this last question depends on p.)