CS 671 (Fall 2020) — Assignment 1

Due: 09/18/2020

If you have not done so yet, read Appendix C, and Chapters 1.0, 1.1, 1.2, 5.2, and 10.2 from the textbook. Here are the homework problems. Recall that you may discuss problems with classmates or ask me for help. If you discuss ideas with classmates, destroy all your notes from your discussion, take an hour off, and then write your own solution. Also, please list on your homework who you collaborated with.

It is *not* appropriate to discuss the problems with people outside the class, or to look for solutions in sources such as published papers, other books, or any online sources. See the course webpage for a more detailed discussion of what you may and may not do.

Homeworks should be typed up (preferably in LaTeX, but this is not required) and submitted as a PDF file (no other formats) by e-mail to David by 5:00pm Pacific on the due date.

(1) In class, we said that we assume "without too much loss of generality" that randomness is provided by a sequence of fair coin tosses. Here, we will revisit this issue.

Suppose that you have some algorithm which needs a uniformly random element of $\{0, 1, 2\}$. But all you have is a stream of uniformly random independent bits. So we need an algorithm which takes as input a (conceptually infinite) sequence of bits, and outputs, after some amount of time, either 0, 1, or 2.

(a) Give (and analyze) a Las Vegas algorithm which always outputs either 0, 1, or 2, and outputs each of them with probability exactly $1/3$. For full credit, your algorithm should in expectation look at no more than $8/3$ bits of the input bit sequence.

(b) Prove that *every* algorithm that always outputs either 0, 1, or 2, and outputs each of them with probability exactly $1/3$, must in expectation look at at least $8/3$ bits of the input bit sequence.

(c) Now suppose that we want a *hard, fixed* bound on the number of bits that your algorithm can look at. In other words, the input sequence has length k, for some finite constant k. You still want an algorithm that always outputs 0, 1, or 2, and outputs each of the three with probability $1/3$. Do one of the following two (preferably not both):

Give an algorithm that always terminates after looking at no more than k bits, and outputs a uniformly random element from $\{0, 1, 2\}$. Prove that your algorithm has the desired properties.

Prove that no algorithm with the desired properties exists.

(2) Suppose that we modified the recursive Fast-Cut algorithm so that if the Min-Cuts of H_1 and H_2 are equally good, the algorithm returns both. (And further up in the recurrence, we might return even more than two Min-Cuts at once.) This would let us find multiple Min-Cuts at once. We'll ignore the issue that the size of the returned objects could be quite large (and thus contributes to the running time).

What fraction of all Min-Cuts would a single run of the modified Fast-Cut algorithm find in expectation? Using this idea, what's the fastest algorithm you can give for finding *all* Min-Cuts of a given graph G? You may pretend that returning an entire cut can somehow magically be done in time $O(1)$.

(3) In the recursive Min-Cut algorithm FastCut, we chose — somewhat arbitrarily — to create two copies of the graph G on which we then ran contractions. Also, we somewhat arbitrarily reduced the size to $n/\sqrt{2}$, to guarantee that with probability at least $1/2$, no edge across the minimum cut was contracted. In this problem, you are asked to generalize the analysis to the algorithm $\text{FastCut}_{k,\alpha}$, which begins by creating $k \geq 2$ copies of the graph G, and then contracts (independently) until each copy has size $\alpha \cdot n$. It then recurses on each of the k copies, and keeps the best solution.

Following along the lines of the analysis in the class/textbook, calculate a bound on the total time that $\text{FastCut}_{k,\alpha}$ takes to guarantee that a minimum cut is found with probability at least $1 - O(n^{-\kappa})$.

Note: this problem is a bit tedious.

(4) For the LP rounding algorithm for MAX-SAT, we rounded each variable independently to true with probability z_i, and to false with probability $1 - z_i$. This gave us a $1 - 1/e$ approximation. While it makes sense to have this rounding probability be increasing in z_i, there is no reason why it should have to be exactly z_i. Here, we will derive a $\frac{3}{4}$ approximation by using probabilities other than z_i themselves.

Specifically, let $g(x)$ be any function satisfying $1 - 4^{-x} \leq g(x) \leq 4^{x-1}$ for all $x \in [0, 1]$.

Prove that if each X_i is set to true independently with probability $g(z_i)$, and to false with probability $1 - g(z_i)$, then the resulting LP rounding is a $\frac{3}{4}$ approximation in expectation.

Give an example of a *linear* function $g(x)$ satisfying the condition (and prove that it does).