CS 671 (Fall 2020) Final Exam (Takehome) — Due 12/02/2020, 5:00pm, by e-mail to David

This exam is takehome, but apart from having more time and getting to work on it wherever you want, you should treat it exactly like an open-book in-class exam. Specifically, here are the key rules. (If in doubt, check with the instructor.)

(a) You may access the textbook, anything I handed out or posted for this class (assignments, sample solutions, whiteboards, lecture videos), and anything you wrote for this class (class notes, homework solutions). You may not access any other books, any online resources (not even Wikipedia), notes taken by classmates or friends, etc.

(b) You may not communicate about this exam with anyone except the instructor. Not with classmates, labmates, friends from undergraduate, or anyone else. Not even basic questions — you should communicate exactly as if you were sitting in an in-class exam, i.e., not at all.

(c) The exam is due Wednesday, 12/02/2020, no later than 5:00pm, as a PDF by e-mail to David. Late submissions will not be accepted.

GOOD LUCK

(1) In class, we used a coupling argument to prove that the random walk on the d-dimensional hypercube mixes in time $O(d \log d)$. Reprove that the random walk mixes rapidly using the technique of canonical paths.

[Note: Probably, the bound you will obtain on the time it takes to be nearly mixed will be worse than the one we obtained using coupling. So long as it is polynomial in d, that is not a problem.]

(2) In class, we talked about sampling colorings of graphs by using a Markov Chain. Another natural type of object to sample is independent sets[1]. Sampling a uniformly random independent set is going to be hard, because the large independent sets could be very far and few between. Instead, we will aim to sample independent sets from a distribution that prefers small independent sets (which are also easier to find). Specifically, for a parameter $\lambda \in (0, 1)$, we will want to sample the independent set S with probability proportional to $\lambda^{|S|}$. Specifically, we will do this only for $\lambda \leq \frac{1}{\Delta}$, where Δ is the maximum degree of the graph G.

We do the sampling with the following Markov Chain: starting from an arbitrary independent set S (the empty set is a natural candidate), we repeatedly perform the following step:

- With probability $\frac{\lambda}{1+\lambda}$, try to add a vertex. To do so, pick a uniformly random vertex v. If $v \in S$ already, or adding v would be illegal, then do nothing. Otherwise, add v to S.
- With the remaining probability $\frac{1}{1+\lambda}$, try to remove a vertex. Again, pick a uniformly random vertex v. If $v \notin S$, then do nothing; otherwise, remove v from S.

(a) Prove that under the stationary probability distribution of this Markov Chain, the chain is at each set S with probability proportional to $\lambda^{|S|}$. (To prove this, it is enough to substitute these values and prove that they are stationary.)

[1] Remember: A vertex set S is independent if no two nodes in it are adjacent.

(b) Use a coupling argument to prove that for $\lambda \leq \frac{1}{\Delta}$, this chain mixes rapidly. (If you prefer, you can use another type of proof — but coupling is likely easiest.) As a hint, you may want to first think about coupling two independent sets S, S' differing in exactly one node, i.e., $S' = S \cup \{v\}$ for some vertex v. If you cannot quite get the proof to work with $\lambda \leq \frac{1}{\Delta}$, you will get most credit for making it work with $\lambda \leq \frac{1}{\Delta+1}$.

[Note: This is probably the hardest question on this exam.]

(3) The Bin Packing problem is defined as follows: you are given n items, with the size of item i being $s_i \in (0,1)$, a fractional number between 0 and 1. You are given a supply of bins, each of size exactly 1. You want to pack all the items in bins, and minimize the number of bins. That is, the objective function is the number of bins used to pack all items. Of course, a bin may be filled only partially, but you cannot overfill any bin. This problem is known to be NP-complete.

(a) Give and analyze a simple deterministic algorithm using at most $2\text{OPT} + 1$ bins, where OPT is the optimum number of bins to use.

(b) We now improve the approximation guarantee to $(1 + 2/e)\text{OPT} + 1$, by using LP-rounding. We say that an n-dimensional 0-1 vector a is *feasible* if the items i with $a_i = 1$ can all fit into one bin, i.e., $\sum_i a_i s_i \leq 1$. Let $a^1, \ldots, a^m$ be all feasible 0-1 vectors. Notice that m will be exponentially large in n. We can now write the following IP for Bin Packing:

$$\begin{array}{lll} \text{Minimize} & \sum_j x_j & \\ \text{subject to} & \sum_j a_i^j x_j \geq 1 & \text{for all } i \\ & x_j \in \{0,1\} & \text{for all } j. \end{array}$$

Of course, we can turn this into an LP in the usual way. The meaning of the variable x_j is that if $x_j = 1$, then the set of items with $a_i^j = 1$ is put into a bin together.

While this LP has exponential size, and is thus not obviously solved, you get to assume that you have a fractional optimum solution x_j. In fact, you get to assume that the solution only has polynomially many x_j entries which are non-zero, and you are given this list of non-zero entries explicitly. So you don't have to deal with exponentially many variables any more.

Show how to use this solution in order to obtain a solution using at most $(1 + 2/e)\text{OPT} + 1$ bins in expectation.

[Hint: you may want to use your solution to part (a).]

(4) Sometimes, randomized algorithms are really good at saving space. Imagine the following scenario: you have a sequence of m items, each from a universe U of size n. Items may repeat, possibly many times, and you want to know how even or uneven the distribution is. Specifically, if item i appears a total of x_i times in the sequence, you want to know the variance $\sum_i (x_i - m/n)^2 = \sum_i x_i^2 - m^2/n$. Since m, n are easy to keep track of, this is equivalent to calculating $\sum_i x_i^2$.

Now, this would be a very easy problem if you could simply count how many times you have seen each element i. Unfortunately, you don't have enough space for n separate counters. You only want to keep track of one number instead. One way to accomplish this — surprisingly — is as follows: for each i, generate a number ξ_i, which is ± 1 with probability $1/2$ each, such that the ξ_i are pairwise independent. Start out with a counter $Y = 0$. Now, whenever you see another occurrence of element i, add ξ_i to Y, i.e., increment or decrement Y, according to the element i.

(a) Prove that $\mathbb{E}\left[Y^2\right] = \sum_i x_i^2$.

(b) The previous part shows that in expectation, this method lets you calculate $\sum_i x_i^2$. Unfortunately, the variance of Y^2 is quite high. To do better, we can run in parallel k independent copies of this algorithm, with independent random choices $\xi_{i,j}$. Call the resulting counters $Y_1, \ldots, Y_k$. Then, we instead keep track of $Z := \frac{1}{k} \cdot \sum_{j=1}^{k} Y_j^2$. Bound the probability that Z deviates from $\sum_i x_i^2$ by more than a $(1 \pm \delta)$ factor, in terms of δ and k.

(c) We motivated this problem by "saving space", yet then, we went ahead and stored n pairwise independent random bits ξ_i. That pretty much defeats the purpose. Show how to use $O(\log n)$ mutually independent random bits to generate n pairwise independent random bits. (Thus, all you really need to store are those $O(\log n)$ bits.)

[Note: You can also get the guarantees from Part (b) with $O(k \log n)$ bits of storage, but that requires techniques that we haven't covered in this class, so you don't need to do that.]

[Hint: The textbook discusses some techniques for generating pairwise independent random numbers from given numbers. I strongly recommend reading that section before attempting this question, but mostly for general intuition — the techniques will likely not be directly applicable here.]