New Theory Frontiers 2024-Feb-14

Had a conversation with Jiayu in the morning, in which we discussed that we would not be working on things for a couple weeks, which gives me some time to think and retread on some of the problems I'm interested in solving. What I need to do is thoroughly go through some basic algorithmic and probabilistic problem-solving techniques from a mathematical standpoint. I've found a few resources towards this goal:

- The book 'Extremal Combinatorics.' I will try to go through the first section of this book with good writing.
- Ryan O'Donnell's Probability and Computing lecture notes from 2009, which are probably worth going through as well.
- Salil Vadhan's course 'A Theorists Toolkit' from 2005, which consists of 12 lectures. Will probably go through them at some point.
- Finally, Babai's notes on 'Linear Algebra methods in Combinatorics.' Will refer to them if I ever need them.

Anyway, I then read chapter 1 of the book on some basic counting. Did the binomial theorem and some definitions of factorials. This is slightly interesting:

Lemma. $\sum_{i=0}^{k} \binom{n}{i} \leq \left(\frac{en}{k}\right)^k$.

Proof uses the identity $1 + t < e^t$. Factorial approximations may be useful; the stirling formula is supposedly important.

$$n! = \left(\frac{n}{e}\right)^n \sqrt{2\pi n} e^{\alpha_n}$$

where $1/(12n+1) < \alpha_n < 1/12n$.

Did the stars and bars counting method and some stuff related to partitions.

Turan's number is quite interesting here, because I remember it being defined only in the context of graphs. An intuitive explanation for this is as follows: the number $T(n,k,l)$ with $(l \leq k \leq n)$ is the smallest number of l-element subsets of an n element set X in a way such that every k-element subset of X contains at least one of those sets. Think of it this way: we have some set X of size n. Then we can divide it into l-element subsets. Suppose we pick a k-element subset; assuming that we have enough l-element subsets, k will necessarily contain one of them. Clearly, if l is simply $\binom{n}{l}$, this is true, but we can choose less of them as well. The smallest number we can choose is $T(l,k,n)$.

Expanders: Benny Applebaum has a lemma in the published version of his PhD thesis which proves the unconditional non-existence

of exponentially secure superlinear stretch PRGs based on an expander result, while his other paper proves the existence of polynomial strech PRGs with superconstant depth and negligible distinguishing advantage based on the plausible random local function assumption (which holds since Goldreich's PRG has been thoroughly attacked and still seems generally plausible). Here are some avenues of attack:

- Can we base the non-existence of PRGs in $\mathbf{NC}^0$ on some kind of reasonable assumption?
- Can we do complicated expander math to prove this contingent on RLF?
- Can we introduce some new expander-esque notion that works for polynomial-stretch PRGs?

Many ways to go about this.

Finished chapter 1 reading. Will do problems tomorrow.

Counting and the Non-Deterministic Time Hierarchy Theorem 2024-Feb-15

I went through the first 15 problems of extremal combinatorics, and spent some time reviewing the material. They were not very hard. The ones after this will probably be more complicated however due to the fact that they require Stirling's approximation.

The meeting with Mike was good; it's clear what we need to do, which is read the papers in depth and try to understand them. It turns out that I've forgotten the proof of the ND Time Hierarchy theorem again. Will take a look at it before I go to bed. I also found out that Cook-Levin is a logspace reduction. I don't know if this is useful or not, but I need to understand it better.

From what I understand, the proof of the theorem is due to this main idea:

First, remember the deterministic time hierarchy theorem. This is very easy: define a deterministic machine D which runs M_x on x given input x for $|x|^{f'(x)}$ steps (where $f'(x) = o(g(x))$ and $f'(x) = \omega(f(x))$), and then it flips the output (or rejects if it doesn't halt in this time). This language is in $\mathbf{DTIME}(g(x))$ because D runs in $g(x)$. However, any machine that runs in f can't work, because provided a long-enough description M_x of such a machine M, it disagrees with D on the input x (since D runs for longer than M and can flip the bit).

This technique does NOT work for NDTMs since 'flipping' is not really a thing: flipping the output on a non-deterministic execution does not work, because there's probably some *other* non-deterministic execution which rejects, but will now be accepted.

The way to get around this is nontrivial. What the machine does is it makes a kind of ridiculous exploit.

Arora-Barak illustrates this using $f = n$ and $g = n^{1.5}$. The idea is basically to define a function $f'(i) = 1, f'(i+1) = 2^{f'(i)^{1.2}}$. That we can find this function in $O(n^{1.5})$ is mildly crazy – it uses the repeated squaring exploit.

But once this is done the machine does the following:

- First, it finds out i given any 1^n where $f'(i) < n < f'(i+1)$. Then it nondeterministically determines what M_i will output on 1^{n+1} and outputs the same thing. Note there's no flipping here.
- Second, if $n = f'(i+1)$ for some i, then it runs M_i on $1^{f'(i)+1}$. The thing is the machine can actually just enumerate over all the non-deterministic workings of M_i. This is because n is actually exponential in $f'(i)+1$. Now it can do this and actually flip the bit.

Immediately, we've solved the (rather horrible) problem. By the first part, $D(1^n) = M_i(1^{n+1})$, because we define the machine so. Now if some machine M_i, which works in f-time can choose the same language as D, the machine automatically gives $M_i(1^n) = D(1^n) = M_i(1^{n+1})$ for all $n < f(i+1)$. Obviously, this is also true for $n = f(i+1)$ and $f(i)+1$. The problem comes with the fact that $D(1^{f(i+1)})$ is **not** equal to $M_i(1^{fi+1})$ by design. So D is strictly stronger than the i's.

Also read up to Lecture 3 of Ryan O'Donnell's notes on probability and computation. Good stuff; nothing that is very new to me, but generally made a lot of sense and I certainly like the fact that they have immediate connections to computer science. Tasks for tomorrow:

- Make the table consisting of all the results, and it's probably time to parse through the 2013 FOCS iO paper to see how they get iO (I wonder how they stylize it? Is it $i\mathcal{O}$? iO? **iO**???)
- Fix some of the comments that Jiayu made on write-up; maybe the nitpicky ones. It's probably not best to try and solve the CDH problem tomorrow.
- Finish the fucking grading. I really wish the students were performing better.
- Do the Stirling approximation problems; at least 2-3 if possible.

The best order is probably 1->3->2->4. I am relying on the power of cumulative intelligence; one year from now I should be smarter

and better. I really hope TT's advice about working hard and 'learning and relearning your field' pays off.