Advanced Algorithm Analysis

Project #2

Introduction: Semirandom Models.

In this unit of the course we'll be starting to introduce some notion of randomness into our investigation; in general, we'll think of this randomness as *external* to the algorithms themselves. I.e., the algorithms we study won't use randomness themselves, but we'll ask questions which use randomness in some meaningful way. For example, we'll perform average-case analysis of algorithms or study what happens when we add some randomness to worst-case inputs.

This project: Your assignment

In collaboration with your work group and with the professor, write 2 problem statements ("deliverables"). Everybody in a work group should complete the same 2 deliverables; this will give provide you with flexibility but also closer working partners as you go through the semester. Each of the deliverables should address a well-posed scientific question and explore some aspect of randomized *analysis* of algorithms. There are lots and lots of options for these, be creative! Here are a couple to get you thinking:

- Take an algorithm that interests you and perform some kind of empirical average-case (runtime, memory usage, approximation ratio) analysis of it, being careful to be scientifically rigorous as you define the distribution which you're investigating.
- Take one of the proofs of the average-case runtime of Quicksort and demonstrate it empirically.
- Take the online selection problem and implement it: could you write an algorithm which takes an arbitrary sequence of probability distributions and then computes the optimal selection policy?
- From the previous point, if you have the optimal online selection algorithm implemented, there are many experiments you could run! For example, you could try keeping the algorithm fixed, but then shuffling the order of the probability distributions, and see how much worse it performs.
- Also with online selection, perhaps you could compare the performance of an optimal online algorithm with the simpler, more robust "prophet" algorithm. I think it might be very interesting to generate experiments which explicitly depict the robustness; for example by shuffling the probability distributions and seeing how this impacts the expected value of the output.

While the project is active, your team will give regular weekly updates in class and I may occasionally ask for demos of the project. When the project is finished, each person in your group will submit a written report which summarizes your findings, and we will supplement the weekly work group update with brief verbal reports from each person.