Problem Set 1 (due Friday, October 30))

Collaboration. You are welcome to collaborate on solving these problems. But you must write the solutions on your own. Please cite any collaborators and/or references that help you in arriving at your solutions.

Problem 1. (30 points) Load balancing

Consider a set of n jobs competing for execution at one of n servers in a cloud computing platform. Each server is capable of executing any job and each job takes one unit of time to complete on any server. We would like to have all the jobs to be completed as soon as possible. Of course, if we assign the ith job to the ith server, then all the jobs can be completed in one unit time. This requires centralized control, and in this exercise we consider the performance of randomized distributed load balancing processes.

(a) Suppose each job is first assigned to a server, independently and uniformly at random. Then, each server executes all the jobs assigned to it in sequence. Show that with probability at least $1 - 1/n$, all the jobs are completed within $O\left(\frac{\log n}{\log \log n}\right)$ steps.

(*Hint:* Use Chernoff bounds. The following version of Chernoff bounds may be more helpful than the versions we covered in class:

Let random variables $X_1, \ldots, X_n$ be independent random variables taking on values 0 or 1. Further, assume that $\Pr[X_i = 1] = p_i$. Then, if $X = \sum_i X_i$ and μ be the expectation of X, then for any $\delta > 0$, we have:

$$\Pr[X > (1+\delta)\mu] < \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^\mu.$$

Note that $(\log n / \log \log n)^{\log n / \log \log n} \leq n$, where the base of log is 2.)

(b) Show that the bound in part (a) is tight to within constant factors. That is, show that with probability at least $1 - 1/n$, there is some job that gets executed after $\Omega\left(\frac{\log n}{\log \log n}\right)$ steps.

(*Hint:* Group the servers into $\sqrt{n}$ groups of $\sqrt{n}$ servers each. Show that each group receives at least $c\sqrt{n}$ jobs with sufficiently high probability, where c is a constant that can be set small enough to achieve a desired probability. Next place a lower bound on the probability that some server is assigned at least $\varepsilon \log n / \log \log n$ jobs – this is the trickiest part of the argument, note that here again you can set the constant ε sufficiently small. Finally, place an upper bound on the probability that none of the servers is assigned at least $\varepsilon \log n / \log \log n$ jobs.)

(c) Consider the following alternative load balancing procedure that repeats the following step until each job has been completed: first each job that has not yet been executed, is tentatively

assigned to a server, independently and uniformly at random; then, each server that has at least one job tentatively assigned to it, executes an arbitrary one of them.

Show that the above load balancing procedure completes all jobs in $\Theta(\log \log n)$ steps with probability at least $1 - 1/n$.

(*Hint:* Determine the expected number of jobs that are completed in the first step. Apply Chernoff bounds for deviation from the mean. Then, determine the expected number of jobs that are completed in the second step, assuming that the deviation is bounded. Repeat this process.)

Problem 2. (15 points) The entropy compression argument

We went through the entropy compression argument of Tao as a way of explaining Moser's constructive proof for Lovasz Local Lemma. (Please refer to link from the class schedule page.) Consider the version of the lemma for finding a satisfiable assignment for a SAT formula in CNF where each clause has exactly k literals and the number of clauses overlapping one clause is at most 2^{k-c}, for some integer constant c. Let m be the number of clauses. The entropy compression argument proceeded as follows.

(a) The number of random bits used in M iterations of Moser's algorithm is exactly $n + Mk$.

(b) A run of the algorithm of length M can be encoded using at most $O(m)+M(k-c+O(1))$ bits in such a way that the encoding together with the n bits that stores the assignment obtained after the M iterations can yield all the random bits used in the run.

Show how to implement the desired encoding of part (b). Your encoding should specify, for each iteration, the unsatisfied clause that was processed in that iteration. Can you suggest a scheme that can encode a run in expected $O(m/2^k) + M(k - c + O(1))$ bits?

Problem 3. (30 points) Probabilistic analysis of a 2-player game

This problem is only an exercise in elementary probability; it does not need any of the bounds we have covered in class. Consider the following two-player game. Initially, the players agree on a pair of positive integers k and n, and Player 0 secretly puts a (uniform) random permutation of the integers 1 through n in an array A (indexed from 1 to n). The game then proceeds in rounds. In round i, $i \geq 1$, Player 0 reveals the *relative order* of the numbers in the first i locations of array A. Player 1 then decides whether to stop the game at round i, or continue to round $i+1$ (if $i = n$, the game must stop). If the game stops at round i, then Player 1 wins iff $A[i] \leq k$.

(a) Prove that if $k = 1$ there is a strategy for Player 1 that wins with constant probability (i.e., with probability at least ε for some positive constant ε).

(b) Prove that for any positive constant c there is a positive constant c' such that if $k \geq c' \log n$ then there is a strategy for Player 1 that wins with probability $1 - n^{-c}$.

(c) Prove that for any positive constant c there is a positive constant c' such that if $k \leq c' \log n$ then no strategy for Player 1 wins with probability $1 - n^{-c}$.

Problem 4. (25 points) Base station locations in a cellular network

Consider the following problem in connection with placing base stations for a cellular network. We are given a set of n locations in a city with distances d_{ij} between locations i and j. For each location j, we are also given a weight w_j that represents the number of customers in location j. We would like to place k base stations in the city (at k of the n locations) so as to serve a maximum number of customers. Each base station can serve customers within a distance of R from the base station.

The problem is to determine k of the locations where to place the base stations so that the total number of customers served is maximized.

(a) **(5 points)** Prove that the decision version of the above problem is NP-complete.

(b) **(5 points)** Express the problem as an integer program.

(c) **(5 points)** Show that a randomized rounding of the linear programming relaxation of the integer program of part (b) yields a solution with expected k base stations and expected number of locations served being at least the optimal.

(d) **(10 points)** Show that a greedy algorithm for the problem (without using LP-relaxations) achieves a constant-factor approximation.