Lecture 1

1 Course Outline

We will cover a sampling of the following topics during the semester.

1. Randomized Load Balancing strategies and their analysis. Probability concepts include, counting, the union bound, and Chernoff bounds.
2. Parallel Algorithms. Here, we discuss basic techniques for parallelizing algorithms in a model that only is concerned with concurrency and not so much with locality.
   (a) PRAM definition. Basic algorithms: list ranking, etc. Reductions to list ranking. In parallel algorithms using a black box algorithm that embodies the parallelism is quite useful. List ranking is one widely useful such black box and an illustration of this concept.
   (b) Definition of NC. P-completeness. Reductions. This is classic complexity as applied to parallelism and gives (conditional) limits on the inherent parallelism available in certain computations.
   (c) Advanced algorithms. Distributed Coloring. Matching. For Distributed Coloring, the task is to break symmetry. The techniques here will also be useful for reducing communication costs as discussed later in the course. The Matching algorithm is beautiful combinatorics and randomized analysis where the influential "Isolation Lemma" was formulated and introduced to computer science.
3. Routing/scheduling
   (a) Butterfly routing: off-line. Benes Routing.
   (b) Valiant routing in the Butterfly network. Randomized strategy for avoiding bad traffic patterns. Bounded queues. To prove results for bounded queues is quite an interesting theoretical endeavor which leads to a somewhat counter intuitive though interesting strategy for switching.
   (c) Path selection in arbitrary networks. (Use linear programming, randomized rounding for approximation.) There is a variant of "the expert's algorithm" here.
   (d) Scheduling in networks. (Lovasz local lemma.) This starts with analysis of problems with scheduling the flow of packets in a networks, includes a discussion of on-line strategies, and also includes the analysis of an approach that eventually yields the "optimal" bounds.

4. Distributed Algorithms

   (a) Byzantine Generals. This shows how to reach agreement in a distributed system in the presense of a rather large fraction of bad agents.

   (b) Algorithms for distributed models. In particular, efficient shortest route computations.

5. Game Theory. Possibilities.

   (a) Nash's Theorem. Proof using Sperner's lemma, and Brower's fixed pt theorem.

   (b) Congestion games. Price of Anarchy. More modern stuff discussing the cost of game theoretic behavior to the utility of a system. (Possible..)

   (c) Arrows theorem. There is no voting methodology for ranking three or more options that acheives three natural properties: no dictator, winner among two is independent of other options, if one is preferred by all it comes first.

   (d) On-line auctions.

6. Partitioning Algorithms.

   (a) Spectral methods. The approach along with intuition about the classical result called Cheeger's inequality which gives one provable bounds on the performance of these methods.

   (b) Linear Programming based methods. Approximation algorithm based on optimization algorithms for graph partitioning.

7. Object Location.

   (a) Distributed Hash Tables. Chord, CANN.

   (b) Locality Based Solutions for special graphs. Tapestry.

   (c) Graph Decomposition based solutions, Approximate shortest path data structures.

2 Load Balancing

You have m jobs to be distributed into n bins. How do you distribute them approximately equally? Duh. Evenly spread the jobs, maximum load is m/n within plus or minus 1.

Ok, let's examine algorithms that use (a lot) less information.

1. Randomly select a bin.

2. Randomly select two bins, choose least loaded.

What is an upper bound on the maximum load for strategy 1? Strategy 2? (We assume for today that $m = n$.)

Isn't n the upper bound!! Oh, yes. Not, what I mean. What bound is the max-load almost always under. Or, with a very small probability, the max-load is greater than what?

A bit of probability...

Probability (for today) is just counting.

We start with an experiment. The experiment gives a sample space which is a set of outcomes along with a set of probabilities for each one. An event is a subset of outcomes.

An example is choosing a poker hand which has the set of poker hands as outcomes where each is chosen with the same probability. An event of interest may be the probability of getting a flush; a poker hand is a 5 card hand and it is a flush when all the cards are of the same suit.

Ok, what is the probability of a flush in a five card poker hand. In this case, we just see what fraction of the outcomes is a flush. The calculation is as follows.

$$\frac{4 * \binom{13}{5}}{\binom{52}{5}}.$$

(Now that we have seen $\binom{n}{k}$, we note that $\binom{n}{k} \leq (ne/k)^k$.)

We will use a couple of other facts repeatedly. For independent events, the probabilities multiply. What is the probability of 4 heads in a row with an unbiased coin?

$$(1/2)^4$$

Back to the calculuation...

So, now that we are experts with probability, let's upper bound the probability that any bin has load greater than k. (We can just enumerate all the situations where any bin has load more than k and divide by n^n. How do we count these situations. Yuck...)

OK, let's just look at one bin. What is the probability that its load is greater than k?

$$\sum_{i \geq k} \binom{n}{i} \left(\frac{1}{n}\right)^i \left(1 - \frac{1}{n}\right)^{n-i} \leq \sum_{i \geq k} \binom{n}{i} \left(\frac{1}{n}\right)^i \leq \sum_{i \geq k} \left(\frac{ne}{i}\right)^i \left(\frac{1}{n}\right)^i = \sum_{i \geq k} \frac{e^i}{i^i}.$$

We can bound this by $2(e/k)^k$. If we choose $k > 2e \log n$, this is bounded by $1/n^2$. So, now we know any one bin is pretty unlikely to have load greater than $2e \log n$.

But, we wanted a bound for all bins, not just for one bin. We use the following simple idea.

The probability of events A or B is at most the probability of A plus the probability of B. Or,

$$Pr[\cup_i A_i] \leq \sum_i Pr[A_i].$$

This is called the union bound, and we use it often in computer science.

If we take A_i to be the event that bin i has load greater than $2e \log n$, we see that the probability of *any* bin has load greater than $2e \log n$ ($Pr[\cup_i A_i]$) is n times the probability than $Pr[A_i]$ which is at most $1/n^2$.

Thus, the probability that *any* bin has load greater than $2e \log n$ is at most $1/n$.

Question 1. Show that with high probability that the max-load is at most $O(\log n / \log \log n)$.

Question 2. Show that expected number of bins with load $\Omega(\log n / \log \log n)$ is at least a constant. (Below is a discussion of expectation, which we will cover in more detail in lecture 3.)

3 A comment about "with high probability."

In the above analysis, appear to have arbitrarily used $1/n$ as an upper bound the probability of "failure." Specifically, I found a value k for which the maximum load was more than k (this is "failure") with probability at most $1/n$.

Why did I choose $1/n$? For example, I could have chosen 1% or .1% or some other number. In general, the notion is that we would like the probability of failure to be small. One notion of small is that it goes to 0. For theorists, we tend to think of n as getting large so $1/n$ tends to go to zero. Other expressions go to zero as well, e.g., $1/\log n$, but somehow $1/n$ is convenient. Moreover, in this case, with respect to the bound on the maximum load k, choosing $1/n$ rather than $1/\log n$ only affects the value of k by a constant factor (actually by a small order term if one were more careful in the analysis.)

4 Expectation: Linearity

We consider a random variable X in some probability space, (which using the above development is defined as a function on a sample space.)

The expectation of the random variable is defined to be

$$E[X] = \sum_{a_i} a_i Pr[X = a_i].$$

For example, in a situation where we throw a ball into one of n bins. Let X be the $0-1$ valued random variable that indicates whether the ball fell into bin 1. It is easy to see that $E[X] = 1/n$ from the definition of expectation. On other hand, using this definition for the situation of throwing n balls into n bins, let's define X as the load on bin 1. What is its expectation? Well,

$$E[X] = \sum_i i * Pr[X = i] = \sum_i i * \binom{n}{i} (1/n)^i (1 - 1/n)^{n-i}.$$

Yuck. We know it is 1. But, the formal reason is *linearity of expectation*, which states that for a random variable $X = Y + Z$,

$$E[X] = E[Y] + E[Z].$$

Now, we can define our random variable X which is the number of balls in bin 1, as the $X = \sum_i X_i$, where X_i is the $0 - 1$ valued random variable that indicates whether ball 1 chose bin 1. Now,

$$E[X] = \sum_i E[X_i] = \sum_i 1/n = 1.$$