CS168, Spring 2015

Mini-Project #1

Due by 11:59 PM on Tuesday, April 7th.

Instructions

You can work individually or with one partner. If you work in a pair, both partners will receive the same grade.

If you've written code to solve a certain part of a problem, or if the part explicitly asks you to implement an algorithm, you must also include your code in your solution to that part. Name your files so that the course staff can easily figure out the contents of each file.

Detailed submission instruction can be found on the course website (`http://cs168.stanford.edu`) under "Coursework - Assignment" section. If you work in pairs, **only one member** should submit all of the relevant files.

Reminder: No late assignments will be accepted, but we will drop your lowest assignment grade when calculating your final grade.

The Power of Two Choices

Goal: The goal of this part of the assignment is to gain an appreciation for the unreasonable effectiveness of simple randomized load balancing, and measure the benefits of some lightweight optimizations.

Description: We consider random processes of the following type: there are N bins, and we throw N balls into them, one by one. We'll compare four different strategies for choosing the bin in which to place a given ball.

1. Select one of the N bins uniformly at random, and place the current ball in it.
2. Select two of the N bins uniformly at random (either with or without replacement), and look at how many balls are already in each. If one bin has strictly fewer balls than the other, place the current ball in that bin. If both bins have the same number of balls, pick of the two at random and place the current ball in it.
3. Same as the previous strategy, except choosing three bins at random rather than two.
4. Select two bins as follows: the first bin is selected uniformly from the first $N/2$ bins, and the second uniformly from the last $N/2$ bins. (You can assume that N is even.) If one bin has strictly fewer balls than the other, place the current ball in that bin. If both bins have the same number of balls, place the current ball (deterministically) in the first of the two bins.

(a) Write code to simulate strategies 1–4. For each strategy, there should be a function that takes the number N of balls and bins as input, simulates a run of the corresponding random process, and outputs the number of balls in the most populated bin (denoted by X below).

(b) Let $N = 100,000$ and simulate each of the four strategies 30 times. For each strategy, plot the histogram of the 30 values of X. Discuss the relative merits of the different strategies. Does one of them stand out as a "sweet spot"?

(c) Propose an analogy between the first of the random processes above and the standard implementation of hashing with chaining (i.e., resolving collisions in a hash table bucket via linked lists). Do the other random processes suggest alternative implementations of hash tables with chaining? Discuss the trade-offs between the different hash table implementations that you propose (e.g., in terms of insertion time vs. worst-case search time).

Deliverables: Your code for part (a); your histograms and discussion for part (b); your answer to part (c).