CS 124

Carlos Peña-Lobel

Colaborated with Christian Hallas

Biased Die Rolls

1. Suppose you are given a six-sided die, that might be biased in an unknown way. Explain how to use die rolls to generate unbiased coin flips, and determine the expected number of die rolls until a coin flip is generated

   Assign probabilities $p_1, p_2, p_3, p_4, p_5, p_6$ to be a, b, c, d, e, f. After 2 dice rolls, we take advantage of the symmetry of ab and ba or any other possible combination. We ignore the difference between 1-4 and a 1-3 for example, and only compare whether or not the first value is greater or less than the second. The 2 rolls will be the same with $p_{same} = a^2 + b^2 + c^2 + c^2 + d^2 + e^2 + f^2$. Now taking the infinite sum, we get an expectation of:

   $$\sum_{i=1}^{\infty} 2i \left(1 - p_{same}\right) \left(p_{same}\right)^{i-1}.$$

   With an actual unbiased die, we can expect to get an unbiased coin flip after 2.4 rolls.

2. Now suppose you want to generate unbiased die rolls (from a six-sided die) given your potentially biased die. Explain how to do this, and again determine the expected number of biased die rolls until an unbiased die roll is generated.

   Following a similar process as above, we now use rolls in sets of 3. This takes advantage of the symmetry of abc, acb, bac, bca, cab, and cba. Since each of these are equally likely we can create a map of each of these rolls to an unbiased die 1-6. With an unbiased die, this occurs in 120/216 of the possible sequences of 3 die rolls, or 5/9 of the time. Taking the same approach as before, we have a much more complicated term for when to reroll, there are ${}_6P_1 = 6$ combos of getting all 3 as the same roll (e.g. $a^3, b^3, c^3...$), and there are ${}_6P_2 = 30$ ways of having a number appear twice (e.g. $a^2b, a^2c, a^2d, ..., b^2a, b^2c, ...f^2d, f^2e$). However, since we must account for order, a^2b can occur in 3 ways: a, a, b; a, b, a; b, a, a, we must multiply each of these terms of order to be 3. Assigning the sum of these two to be denoted as $p_{duplicate}$, we can write an expectation as:

   $$\sum_{i=1}^{\infty} 3i \left(1 - p_{duplicate}\right) \left(p_{duplicate}\right)^{i-1}.$$

   Using this method with an actual unbiased die, we would expect there to be an unbiased roll after 5.4 rolls.

Fibonacci

1. On a platform of your choice, implement the three different methods for computing the Fibonacci numbers (recursive, iterative, and matrix) discussed in lecture. Use integer variables. How fast does each method appear to be? Give precise timings if possible.

2. Modify your programs so that they return the Fibonacci numbers modulo $65536 = 2^{16}$. For each method, what is the largest value of k such that you can compute the kth Fibonacci number (or the [kth Fibonacci number] modulo 65536) in one minute of machine time?

   See code for this answer. Python doesn't have integer integer overflow, so I ran an increasingly finer grid search overnight (when there is supposedly less background programs running) to compute the largest Fibonacci number that could be computed in one minute of machine time. This was done for all 3 methods, using both the moded version and the unmoded version.

   Recursive Function: $40 \rightarrow 40$

   Iterative Function: $352,019 \rightarrow 157,365,989$

   Matrix Function: $19,489,354 \rightarrow$Largest number tested was $\approx 10^{200}$ and ran in 40ms, see github

   I have attached the ipython notebook file I used, a python file of the .ipynb, and here is a link to my github with print outs of the timing:

Run Time Notation

1. Indicate for each pair of expressions (A, B) in the table below the relationship between A and B. Your answer should be in the form of a table with a "yes" or "no" written in each box. For example, if A is $O(B)$, then you should put a "yes" in the first box.

A	B	O	o	Ω	ω	Θ
$\log n$	$\log(n^2)$	✓		✓		✓
$\log(n!)$	$\log(n^n)$	✓		✓		✓
$\sqrt[3]{n}$	$(\log n)^6$			✓	✓	
$n^2 2^n$	3^n	✓	✓			
$(n^2)!$	n^n			✓	✓	
$\frac{n^2}{\log n}$	$n \log(n^2)$			✓	✓	
$(\log n)^{\log n}$	$\frac{n}{\log(n)}$			✓	✓	
$100n + \log n$	$(\log n)^3 + n$	✓		✓		✓

Proofs

For all of the problems below, when asked to give an example, you should give a function mapping positive integers to positive integers. (No cheating with 0's!)

1. Find (with proof) a function f_1 such that $f_1(2n)$ is $O(f_1(n))$.

Choose $f_1(n) = n$, then we have $f_1(2n) = 2n$. By choosing $c = 3$, and N to be any positive integer (eg 1), it follows that $f_1(2n) \leq 3f_1(n)$ for any $n \geq N$.

2. Find (with proof) a function f_2 such that $f_2(2n)$ is not $O(f_2(n))$.

   Choose $f_2(n) = 2^n$, then we have $f_2(2n) = 2^{2n}$. Rewriting 2^{2n} as $2^n 2^n$ or $2^n f_2(n)$ shows that in order for $f_2(2n)$ to be $O(f_2(n))$, we would need to be able to write, $2^n f_2(n) \leq c f_2(n)$. However, 2^n has no upper bound, while c must be a fixed number, so for every c, it is impossible to choose an N s.t. $n \geq N$.

3. Prove that if $f(n)$ is $O(g(n))$, and $g(n)$ is $O(h(n))$, then $f(n)$ is $O(h(n))$.

   If $f(n)$ is $O(g(n))$, this means we can choose a c_1 and N_1 such that $\forall\, n \geq N_1, f(n) \leq c_1 g(n)$. Likewise, if $g(n)$ is $O(h(n))$, this means we can choose a c_2 and N_2 such that $\forall\, n \geq N_2, g(n) \leq c_2 h(n)$. We can now choose $N_3 = max\{N_1, N_2\}$, and since a condition is that $\forall\, n \geq N$ our equality must hold, our new N_3 must also hold from the inequality because it $N_3 \geq N_2, N_1$. We can also choose $c_3 = c_1 c_2$. Now we know that $f(n) \leq c_1 g(n) \leq c_1 c_2 h(n) = c_3 h(n)\ \forall\, n \geq N_3$, and thus $f(n)$ is $O(h(n))$.

4. Give a proof or a counterexample: if f is not $O(g)$, then g is $O(f)$.

   A counter example. Let $f(n) = n^2$, and let $g(n) = n^2(sin(x) + 1)$.

5. Give a proof or a counterexample: if f is $o(g)$, then f is $O(g)$.

   Using the definition of $o(g(n))$ we have

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0.$$

   This tells us that $\exists$ a sufficiently large N, s.t. $\forall\, n \geq N, f(n) \leq g(n)$. This allows us to choose a constant c as 1, because this inequality is already stronger than that of the definition of $O(g(n))$.