CS591B1: Answers to Quiz **Spring 2002**

Lecture 8 — March 25, 2002

Lecturer: John Byers *BOSTON UNIVERSITY* *Scribe: Jeffrey Considine*

1. Order the entropy of the event spaces below from highest to lowest (do not calculate explicitly!). Give a brief reason for each of the inequalities in your ordering.

   (A) $H(\frac{1}{777}, \frac{1}{777}, \frac{1}{777}, \ldots, \frac{1}{777})$
   (B) $H(\frac{1}{775}, \frac{1}{775}, \frac{1}{775}, \ldots, \frac{1}{775})$
   (C) $H(\frac{2}{777}, \frac{2}{777}, \frac{1}{777}, \ldots, \frac{1}{777})$

   Answer: $(A) > (B) > (C)$. $(A) > (B)$ because they are both uniform partitions and (B) has fewer partitions. $(B) > (C)$ because they both have the same number of partitions and the uniform one has more entropy.

2. Suppose you build an erasure-resilient transmission that just divides a file up into n pieces and sends those n pieces in round-robin order until a client has received all the pieces (i.e. no "coding" is used). If each piece is lost independently at random with probability 50%, show that the expected decoding inefficiency of this scheme is $\Theta(\log n)$.

   The simplest way to approach this problem is to define a recurrence such as $T(n) = 1+T(n/2)$ on the *expected* number of rounds needed to receive n distinct packets. The real recurrence is $T(n) = \sum \frac{\binom{n}{i}*T(i)}{2^n}$.

   Answer: Consider $P_c(n)$, the probability that all n pieces of the file have been received after $c\lg n$ rounds.

$$\begin{aligned}
P_c(n) &= \left(1 - \left(\frac{1}{2}\right)^{c\lg n}\right)^n \\
&= \left(1 - \frac{1}{n^c}\right)^n \\
&= \left(1 - \frac{1}{n^c}\right)^{n^c * n^{1-c}} \\
\left(\frac{1}{e} * \left(1 - \frac{1}{n^c}\right)\right)^{n^{1-c}} &\leq P_c(n) \leq \left(\frac{1}{e}\right)^{n^{1-c}}
\end{aligned}$$

The upper bound implies that for $c < 1$, $c \lg n$ rounds fail to recover all pieces with high probability. This suffices to show that the decoding inefficiency is $\Omega(\lg n)$. Actually, a constant probability of failure would have sufficed.

For the upper bound, we need a more managable lower bound on $P_c(n)$.

$$\begin{aligned}
\left(1 - \frac{1}{n^c}\right)^{n^c} &\leq \frac{1}{e} \\
&\leq \left(\frac{1}{e}\left(1 - \frac{1}{n^{c+2}}\right)\right)^{n^{-1}} \\
&= \left(\left(\frac{1}{e}\left(1 - \frac{1}{n^{c+2}}\right)\right)^{n^{1-(c+2)}}\right)^{n^c} \\
&\leq P_{c+2}(n)^{n^c} \\
1 - \frac{1}{n^c} &\leq P_{c+2}(n) \\
P_c(n) &\geq 1 - \frac{1}{n^{c-2}}
\end{aligned}$$

This demonstrates that (a loose upper bound on) the probability of failure drops very quickly after $2 \lg n$ rounds. Using this fact, we can derive the upper bound on the expected number of rounds.

$$\begin{aligned}
T(n) &\leq \int_0^\infty c \lg n * (1 - P(c)) dc \\
&\leq \lg n * \int_0^\infty c * (1 - P(c)) dc \\
&\leq \lg n * \int_0^\infty c * \frac{1}{n^{c-2}} dc \\
&= \lg n * O(1) \\
&= O(\lg n)
\end{aligned}$$

3. Section 4.4 in the scribe notes lists some design improvements for CANs. Which of these are applicable to Chord and why?

Answer: Multi-dimensioned coordinate space This improvement doesn't really apply since Chord is strictly one-dimensional. It would be easier to start from CAN and add finger tables in place of a single pointer to a neighbor in each dimension.

Realities: multiple coordinate spaces This could be done with the same benefits.

Better CAN routing metrics This wouldn't be very good - if each hop took a "better" entry instead of the longest that didn't overshoot, then the $O(\log n)$ hops could easily turn it into $\Omega(n)$. Appropriate weights (by length of hop) would make it very difficult to find a better entry.

Overloading coordinate zones Reducing the number of zones would only reduce the number of hops by an additive constant (as opposed to a constant *factor* for CAN). There would be routing benefits (latency-wise) and better fault tolerance. Overall, it would help but not as much as for CAN.

Multiple hash functions This would work and get an expected path length reduction of $\lg k$ (additive).

Topologically-sensitive construction of the CAN overlay network Big jumps from finger tables make this difficult to make it work, but it would help if it done right.

More Uniform Partitioning This would benefit in the same way as CAN.

4. Suppose m balls are thrown into n bins uniformly at random.

(a) Derive a formula for the expected number of empty bins as a function of m and n.

Answer: Let $X_i = 1$ if bin i is empty. Let X be the number of empty bins. Clearly, $X = \sum X_i$.

$$\begin{array}{rcll} E[X] & = & E[\sum X_i] & \text{substitution} \\ & = & \sum E[X_i] & \text{linearity of expectations} \\ & = & n * E[X_i] & \text{there are } n \text{ bins} \\ E[X_i] & = & Pr[X_i = 1] & \text{Bernoulli variable} \\ & = & \left(\frac{n-1}{n}\right)^m & \text{probability of falling in another bin } m \text{ times} \\ & = & \left(1 - \frac{1}{n}\right)^m & \\ & \approx & e^{-m/n} & \\ E[X] & = & n * \left(1 - \frac{1}{n}\right)^m & \textit{substitution} \\ & \approx & n * e^{-m/n} & \end{array}$$

Intuition:

For intuition about how linearity of expectations is applicable since the X_i's are clearly dependent, consider the following table.

Event	*Bins*			
	Bin 1	*Bin 2*	*Bin 3*	...
e_1	0	0	0	0
e_2	0	0	0	1
e_3	0	0	1	0
e_4	0	0	1	1
...				

Each row is a way the balls fell in bins. Each column is all the possibilities for a particular X_i. There are associated weights (probabilities) for each event which are not shown. $E[\sum X_i]$ is a weighted sum over all rows of this table of all the X_i's of those rows. $\sum E[X_i]$ is a sum over all columns of the weighted sums of the X_i's of those columns. Since the weights are the same in both cases and only the order of traversal changed, the results are the same.

(b) Now suppose $m = 2n$. Give an upper bound on the probability that bin j receives more than 10 balls.

Answer: Let X be the number of balls in bin j. Noting that the expected number of balls in bin j is $m/n = 2$ and applying Chernoff bounds,

$$\begin{aligned} Pr[X > 10] &= Pr[X > (1+4) * 2] \\ &< \left(\frac{e^4}{(1+4)^{1+4}}\right)^2 \end{aligned}$$

5. Suppose your friend tells you he has a way to augment a Bloom filter to check for false positives – in particular, he claims that by tripling the size of a conventional Bloom filter, he can both avoid false negatives (as before) and if the answer is positive, he can tell whether or not it is a false positive. Sketch how to build such an augmented filter or prove that this is impossible.

   Answer: This is impossible. Given such a data structure, one could exactly reconstruct the set of elements inserted by testing all elements and throwing outthe false positives. This requires $\Omega(n \log |U|)$ bits for n elements in universe U but the Bloom filter (and the alleged improvement) use only $O(n)$ bits.

6. How do Tornado and Reed-Solomon codes differ? What are the pros and cons in regards to how effectively they approximate a Digital Fountain?

   Answer: Tornado codes are faster, running in expected linear time, have randomized behavior and generally require more output symbols. Reed Solomon codes always need exactly n distinct symbols to recover n inputs but are slow, running in quadratic time or $O(n \log n)$ with a very large constant. The areas of difference are therefore speed of encoding and decoding, the number of output symbols required and to a lesser extent, predictability.

   Ignoring execution time, Reed-Solomon codes can produce a good digital fountain. Tornado codes are only an approximation since they need the extra symbols but the speed advantage usually makes them more practical. One practical issue both codes face is the fixed stretch factor which can hinder the uniqueness of each symbol in the digital fountain if the stream repeats.

7. (a) Suppose your handler informs you that a secret message is the constant coefficient C in a quadratic equation of the form: $P(x) = Ax^2+Bx+C$, where computations are over Z_{119}. Suppose you then acquire the following $(x, P(x))$ pairs: (1, 4), (4, 25), (5, 36) and (10, 2). What is the message?

   Answer: The equation happens to be $P(x) = (x+1)^2 = x^2 + 2x + 1$ so $C = 1$. This can be found with Lagrangian interpolation.

   (b) How many unique 3rd degree equations (i.e $Ax^3 + Bx^2 + Cx + D = 0$) with coefficients in Z_{119} pass through the four pairs you acquired in part a? How many unique 4th degree equations pass through the four pairs? (You don't have to find them – just specify how many).

   Answer: *This answer ignores the fact that* $119 = 7 * 17$ *so* Z_{119} *doesn't really form a field.* There is exactly one 3rd degree equation since there are 4 points specified. There are 119 4th degree equations - any unspecified $P(x)$ value (i.e. for $x \notin \{1, 4, 5, 10\}$) has 119 possibilities, one for each value in the field. Choosing a $P(x)$ value for one more x precisely determines the equation.