CMSC498K Homework 4 Solutions

Richard Matthew McCutchen

Problem 1

Let T be the event that the coin lands tails, and let W be the event that the selected ball is white. The coin is fair, so $P(T) = 1/2$. If the coin lands tails, the ball is taken from urn B, which has 3 white balls among a total of 15, so $P(W \mid T) = 3/15$. Similarly, $P(W \mid \neg T) = 5/12$. We want to find $P(T \mid W)$. By Bayes's Law:

$$P(T \mid W) = \frac{P(T \cap W)}{P(W)} = \frac{P(T) \cdot P(W \mid T)}{P(T) \cdot P(W \mid T) + P(\neg T) \cdot P(W \mid \neg T)} = \frac{(1/2)(3/15)}{(1/2)(3/15) + (1/2)(5/12)} = 12/37.$$

Problem 2

Let $\sigma_i(S)$ denote the element of S that comes first in σ_i; then the sketch of S is $\big(\sigma_i(S)\big)_{i=1}^{k}$. Consider a fixed i. σ_i is random, so $\sigma_i(A \cup B)$ is equally likely to be any of the elements of $A \cup B$, each with probability $1/|A \cup B|$. In particular,

$$P\big(\sigma_i(A \cup B) \in A \cap B\big) = |A \cap B|/|A \cup B| = s(A, B).$$

Now observe that $\sigma_i(A \cup B) \in A \cap B$ if and only if $\sigma_i(A) = \sigma_i(B)$. For if $\sigma_i(A \cup B)$ is in $A \cap B$, then it is in both A and B, while neither A nor B has an element that comes before it in σ_i; thus $\sigma_i(A) = \sigma_i(A \cup B) = \sigma_i(B)$. Conversely, if $\sigma_i(A) = \sigma_i(B) = x$, then x is in both A and B and neither A nor B has an element that comes before x in σ_i, so $\sigma_i(A \cup B) = x \in A \cap B$. Therefore, $P\big(\sigma_i(A) = \sigma_i(B)\big) = s(A, B)$.

For each i, let X_i be a random variable that is 1 if $\sigma_i(A) = \sigma_i(B)$ and 0 otherwise. We have shown that $E(X_i) = s(A, B)$. Let $X = \sum_{i=1}^{k} X_i$; then $E(X) = ks(A, B)$. The orderings σ_i are all independent, so the variables X_i are all independent, so we can apply the two-sided Chernoff bound to X:

$$\begin{aligned}
& P\big(|X - E(X)| \geq \epsilon E(X)\big) && \leq && 2\exp(-\epsilon^2 E(X)/3) \\
\Rightarrow \quad & P\big((1-\epsilon)ks(A,B) \leq X \leq (1+\epsilon)ks(A,B)\big) && \geq && 1 - 2\exp(-\epsilon^2 ks(A,B)/3) \\
\Rightarrow \quad & P\big((1-\epsilon)s(A,B) \leq X/k \leq (1+\epsilon)s(A,B)\big) && \geq && 1 - 2\exp(-\epsilon^2 ks(A,B)/3)
\end{aligned}$$

Thus, X/k is a good estimate of $s(A, B)$, and we can compute it easily by comparing the two sketches. Suppose we want

$$(1-\epsilon)s(A,B) \leq X/k \leq (1+\epsilon)s(A,B)$$

to hold with error probability δ. It is sufficient that:

$$\begin{aligned}
2\exp(-\epsilon^2 ks(A,B)/3) &\leq \delta \\
\Leftrightarrow \quad 2/\delta &\leq \exp(\epsilon^2 ks(A,B)/3) \\
\Leftrightarrow \quad \ln(2/\delta) &\leq \epsilon^2 ks(A,B)/3 \\
\Leftrightarrow \quad k &\geq \frac{3\ln(2/\delta)}{\epsilon^2 s(A,B)}.
\end{aligned}$$

Note: It occurs to me that Chernoff bounds are a form of statistical inference and thus the resulting bounds should be taken with the same grain of salt as traditional confidence intervals. Consider an experiment that

measures a statistic $\hat{x}$ and uses it as an estimate of a parameter x having some prior probability distribution. Statistical inference gives the probability that $\hat{x}$ estimates x well when x has a particular value, or when weighted by the prior probability distribution of x, the overall probability that $\hat{x}$ estimates x well whatever x is. These are not the same as the probability that $\hat{x}$ estimates x well *given the observed value of* $\hat{x}$, which is the relevant probability in an experiment. Still, statistical inference is a reasonable way for computer scientists to analyze a statistical algorithm without reference to a particular prior probability distribution.

Problem 3

This is easily done with a hash function. Current computers make it most convenient to use something like MD5 or SHA-1, but we'll use a universal hash function to get a provable probability bound. Before departure, the explorers should agree on a positive integer m, a prime $p > \max(m, 2^n)$, and a randomly chosen hash function $h \in \mathcal{H}_{p,m}$. When the explorers reach their respective planets, the first explorer interprets the DNA of the Mars species as an integer x between 0 and $2^n - 1$, computes $h(x)$, and sends it to the second explorer. The second explorer interprets the DNA of the Venus species as an integer y in the same way, computes $h(y)$, and declares the two species identical if and only if $h(x) = h(y)$.

If the species are actually identical, then $x = y$, so $h(x) = h(y)$ and the explorers will correctly determine this. If the species differ, then it is a property of the universal hash function $\mathcal{H}_{p,m}$ that $P_h\big(h(x) = h(y)\big) \leq 1/m$; thus, the explorers will determine that the species differ with error probability $1/m$. The explorers can make the error probability as small as desired by choosing large m.

Problem 4

Consider a graph $G = (V, E)$. Here's one easy algorithm:

- Load all edges into a hash table for constant-time adjacency checks.
- For each edge $e_1 = (a, b)$ and each edge $e_2 = (c, d)$:
  - Check whether the edges (a, c), (a, d), (b, c), and (b, d) are all present. If so, conclude that the graph has the K_4 $\{a, b, c, d\}$.
- If no K_4s were found, conclude that the graph doesn't have any.

Correctness should be obvious. There are $O(|E|^2)$ choices of e_1 and e_2 and it takes constant time to test each one, so the algorithm runs in $O(|E|^2)$ time.

We can do better if G is d-inductive:

- Load all edges into a hash table.
- Find a d-inductive order of the vertices by repeatedly deleting the lowest-degree vertex, and arrange the vertices in order from last deleted on the left to first deleted on the right. Scan the edges and build a "left adjacency list" L_v for each vertex v containing its neighbors to the left (of which there are at most d). This procedure was discussed in class on February 26.
- For each edge (c, d) (suppose c is left of d):
  - Construct $L_c \cap L_d$ by scanning L_c and checking whether each vertex is also a left neighbor of d.
  - For each pair of vertices $a, b \in L_c \cap L_d$, check whether (a, b) is an edge. If so, conclude that the graph has the K_4 $\{a, b, c, d\}$.
- If no K_4s were found, conclude that the graph doesn't have any.

If the algorithm reports that $\{a, b, c, d\}$ forms a K_4, it is correct because it ensured that (c, d) and (a, b) are edges and that both a and b are (left) neighbors of both c and d. Furthermore, if there is a K_4, the algorithm will consider its two rightmost vertices as c and d and its two leftmost vertices as a and b and thereby find the K_4. Thus, the algorithm is correct.

For the running time: The greedy deletion procedure can be done in $O(|E|)$ total time if the lowest-degree vertex is found using the same data structure as in the constant-worst-case candidate heavy-hitters algorithm. Constructing the hash table and all the left adjacency lists takes $O(|E|)$ time. The main loop considers $|E|$ edges (c, d) and processes each in $O(d^2)$ time since $|L_c \cap L_d| \leq d$. Thus, the total running time is $O(d^2|E|)$. It is also $O(d^3|V|)$ since $|E| \leq d|V|$ for a d-inductive graph.