CS 256: Algorithm Design and Analysis

Assignment 9 (due 12/03/2020)

Instructor: Sam McCauley

Problem 1 (KT 13.2)**.** Consider a county in which 100,000 people vote in an election. There are only two candidates on the ballot: a Democratic candidate (denoted D) and a Republican candidate (denoted R). As it happens, this county is heavily Democratic, so 80,000 people go to the polls with the intention of voting for D, and 20,000 go to the polls with the intention of voting for R.

However, the layout of the ballot is a little confusing, so each voter, independently and with probability 1/100, votes for the wrong candidate—that is, the one that he or she *didn't* intend to vote for. (Remember that in this election, there are only two candidates on the ballot.)

Let X denote the random variable equal to the number of votes received by the Democratic candidate D, when the voting is conducted with this process of error. Determine the expected value of X.

Solution. □

Problem 2. Consider the process of throwing n balls into m bins, where each ball is thrown into a uniformly random bin, independent of other balls. What is the expected number of balls in a particular bin b? (*Hint.* Define appropriate indicator random variables and use linearity of expectation.)

Number of collisions in a hash table with chaining. This analysis gives, under a random hash function, the expected number of collisions in a hash table with chaining if n items are stored in a hash table with m slots.

Solution. $\square$

Problem 3 (Erickson handout)**.** Your boss wants you to find a perfect hash function for mapping a known set of n items into a table of size m. A hash function is perfect if there are no collisions; each of the n items is mapped to a different slot in the hash table. Of course, a perfect hash function is only possible if $m \geq n$.

After cursing your algorithms instructor for not teaching you about (this kind of) perfect hashing, you decide to try something simple: repeatedly pick random hash functions until you find one that happens to be perfect.

(a) Suppose you pick an random hash function h. What is the exact expected number of collisions, as a function of n (the number of items) and m (the size of the table)?

Dont worry about how to resolve collisions; just count them.

(b) What is the exact probability that a random hash function is perfect?

(c) What is the expected number of different random hash functions you have to test before you find a perfect hash function? Give an exact answer (using your answer to part (b)), then simplify your answer using the techniques we've seen in class. (The simplified answer should be a function of m and n, rather than a product of many terms.)

Solution. □

Problem 4. In class, we saw simple randomized algorithms that give a constant factor approximation to NP hard problems like MAX-3-SAT and Max-Cut. In this question, we design a simple deterministic approximation algorithm for the NP hard problem, Vertex Cover. Consider the following simple strategy:

Start with vertex cover $S \leftarrow \emptyset$
While there is an uncovered edge $e = (u, v)$:

- Add both endpoints to S, that is, $S \leftarrow S \cup \{u, v\}$
- Delete all edges that are incident on u and v

Show that the above algorithm is a 2-approximation. In particular, show that S is a vertex cover and that $|S| \leq 2 \cdot |S^*|$, where S^* is a optimal (minimum-size) vertex cover.

Solution. □