CS 6150: HW5 – Randomized algorithms

Submission deadline: Friday, Nov 18, 2016, 11:59PM

This assignment has 5 questions, for a total of 50 points. Unless otherwise specified, complete and reasoned arguments will be expected for all answers.

Question	Points	Score
Balls and Bins	15	
Estimating the Mean and Median	10	
Quick-sort with Optimal Comparisons	10	
Randomized Min-Cut	10	
Valiant-Vazirani Lemma	5	
Total:	50	

Question 1: Balls and Bins ..... **[15]**

For the purposes of this exercise, $\log n$ denotes the natural logarithm.

(a) **[3]** Suppose we have $m = 4n \log n$ balls, and we throw them into n bins (choosing a bin uniformly at random for each ball, as we saw in class). Prove that the probability that there exists an empty bin is $< 1/n$.

(b) **[3]** What can you say about the probability above, when (a) $m = \frac{1}{2} \cdot n \log n$, and (b) $m = 100n \log n$?

(c) **[2]** Let us now go back to the setting of $m = n$. We saw in class that the probability that bin j is empty is $\left(1 - \frac{1}{n}\right)^n \leq 1/e$ (and the quantity tends to $1/e$ as $n \to \infty$). Use Markov's inequality to bound the probability that 90% of the bins are empty.

(d) **[7]** Let us now obtain a much better bound. Let X_j denote the random variable that is 1 if bin j is empty, and 0 otherwise. We discussed in class why the X_j's are *not* independent. Thus, we cannot use the Chernoff bound to bound the probability that 90% of the bins are empty.

However, the variables in this case are "anti-correlated". Formally, prove that for any distinct indices $j_1, j_2, \ldots, j_k$, we have

$$\Pr[X_{j_1} = 1 | X_{j_2} = X_{j_3} = \cdots = X_{j_k} = 1] \leq \Pr[X_{j_1} = 1].$$

I.e., conditioning on other bins being empty only decreases the probability that a given bin is empty. Use this to prove that the probability that 90% of the bins are empty is at most $(0.9)^n$ (which is exponentially small in n).

[*Hint:* first use the above inequality along with Bayes rule to prove that $\Pr[X_{j_1} = X_{j_2} = X_{j_3} = \cdots = X_{j_k} = 1] \leq e^{-k}$.]

Question 2: Estimating the Mean and Median ..... **[10]**

The point of this exercise is to learn to *apply* Chernoff bounds in simple cases.

(a) **[3]** Let $a_1, a_2, \ldots, a_n$ be n real numbers in $[-1, 1]$. Say we would like to compute their mean by sampling indices j at random from $\{1, 2, \ldots, n\}$ (with replacement) and computing the average of the sampled a_j's. How many indices must we sample in order to obtain an estimate $\widehat{\mu}$ of the mean μ such that $|\widehat{\mu} - \mu| \leq \epsilon$ with probability $1 - \delta$?

[*Hint:* You may want to go through the survey on Concentration bounds by Chung and Lu, linked on the course page.]

(b) **[1]** Suppose we sample without replacement. Does your proof above still work? (Answer yes/no, with a couple of lines of reasoning.)

(c) **[2]** Suppose we weaken the constraint on a_i to $a_i \in [-M, M]$, for some parameter M. How many samples must we now take in order to obtain an estimate $\widehat{\mu}$ with the same guarantee as in part (a)?

(d) **[4]** Suppose n is odd. Prove that it is not possible to obtain an estimate of the *median* via samples. Formally, show that if we have $a_1, a_2, \ldots, a_n \in [0, 1]$, and given an $\epsilon < 1/8$, it is not possible to estimate the median of the a_i's up to a $\pm\epsilon$ error by sampling $o(n)$ of the a_i's (with or without replacement – it does not matter) and computing the sample median.

Question 3: Quick-sort with Optimal Comparisons ..... **[10]**

We have seen in one of the lectures that *any* comparison-based sorting algorithm must make at least $n \log_2 n - O(n)$ comparisons in the worst case, when given an array of size n. Now we will give a randomized algorithm that achieves this bound asymptotically (in expectation).

Let $A[0, \ldots, n-1]$ be the input array. Consider a variant of quick-sort, in which instead of picking a uniformly random pivot, we sample M random elements of A (without replacement), and pick the median of these entries as a pivot. The intuition is that this will lead to a near-perfect split with good probability, and thus the number of comparisons will follow the recurrence that "nearly" looks like $T(n) = 2T(n/2) + n$.

(a) **[3]** Suppose $M = 2m + 1$, for an integer $m \geq 1$. Let p_k be the probability that the k'th smallest element in A is chosen as the pivot by the above procedure ($1 \leq k \leq n$). Prove that

$$p_k = \frac{\binom{k-1}{m}\binom{n-k}{m}}{\binom{n}{2m+1}}.$$

(b) **[3]** Give a recursive formula for the expected number of comparisons on an array of size n, in terms of the numbers p_k defined above.

(c) **[4]** Now, we can use an approximation for p_k to solve the recursion. It turns out that we can approximate

$$p_k \approx \frac{(2m+1)!}{m!\ m!} \cdot \frac{1}{n} \left(\frac{k}{n}\right)^m \left(1 - \frac{k}{n}\right)^m,$$

and plug it into the recursion above. For $m = 1$ and $m = 5$, prove by induction that the expected number of comparisons is $\leq C_m \cdot n \log_2 n$, for appropriate constants. You get half credit if $C_5 < C_1$, and full credit if additionally, $C_5 \leq 1.6$. [*Hint:* approximate the summation with an integral,[1] and use Wolfram Alpha to obtain numeric approximations.]

Question 4: Randomized Min-Cut ..... **[10]**

The point of this exercise is to work out the details of the very simple, randomized algorithm for 'global min-cut', proposed by Karger. As this is now standard in randomized algorithms classes, you can easily find the solution online, but I encourage you to work out the calculations by yourself first.

The global min cut problem is the following: given a connected undirected graph $G = (V, E)$ (no edge weights), the goal is to remove as few edges as possible so as to disconnect the graph.

Karger's algorithm is extremely simple: at every step, pick a random edge in the graph (uniformly at random), and "collapse" its end points into a supernode (this gives a new graph with one vertex less, on which we recurse). In the process, we keep the parallel edges, and discard self loops.

The algorithm ends when precisely two vertices remain, at which point we return the number of (parallel) edges between the vertices.

(a) **[2]** Let E' be one of the min cuts in the graph (it is the subset of edges that need to be cut). Prove that if we collapse an edge that is *not* in E', then the size of the min cut in the new graph is equal to that in G.

(b) **[2]** If E' is one of the min cuts in the graph, prove that $|E'| \leq 2|E|/n$. [*Hint:* what is the average number of edges in a 'vertex cut'? (a vertex cut is a cut in which we remove all the edges incident on one vertex)]

(c) **[2]** Use these observations to prove that the probability that the min cut value is maintained after one step is at least $\left(1 - \frac{2}{n}\right)$.

We can then recurse, and obtain that the overall probability of success is at least

$$\left(1 - \frac{2}{n}\right)\left(1 - \frac{2}{n-1}\right) \ldots \left(1 - \frac{2}{3}\right) \approx \frac{2}{n^2}.$$

Thus, if we run the algorithm $O(n^2)$ times, there is a very high probability that at least one of the times, the process *succeeds* in finding the cut E'.

(d) **[4]** One surprising consequence of the algorithm is that the *number* of min cuts in an unweighted graph is small. Prove, using the last observation above, that the number of min cuts is $\leq n^2/2$. (Significantly smaller than the total number of cuts, which is 2^n.)

[*Hint:* Suppose we have min cuts $E'_1, E'_2, \ldots E'_r$. We could have used the reasoning in parts (a)-(c) with any of the min cuts, and we have that the probability that we end up with E'_i is at least $2/n^2$ for each i.]

[1] For those who haven't seen this trick, please talk to the TA's for explanation. (Or Google for good resources, and share!)

Question 5: Valiant-Vazirani Lemma ..... **[5]**

Let $a_1, a_2, \ldots, a_m$ be random integers chosen independently from the interval $[1, N]$. Prove that the probability that the "argmin" of the a_i (i.e., the index j s.t. a_j is minimum) is unique with probability at least $\left(1 - \frac{1}{N}\right)^{m-1}$.

[*Remark:* This is a special case of a general and powerful result known as the *Isolation Lemma* (see the Wikipedia article on the same for more details).]

[*Hint:* Prove by induction on m.]