COMPSCI 690RA: Problem Set 2

Due: 3/3 by 8pm in Gradescope.

Instructions:

- You are allowed to, and highly encouraged to, work on this problem set in a group of up to three members.
- Each group should **submit a single solution set**: one member should upload a pdf to Gradescope, marking the other members as part of their group in Gradescope.
- You may talk to members of other groups at a high level about the problems but **not work through the solutions in detail together**.
- You must show your work/derive any answers as part of the solutions to receive full credit.

Hint: The following two inequalities may be helpful throughout the course: for any $x > 0$, $(1 + x)^{1/x} \leq e$ and $(1 - x)^{1/x} \leq 1/e$.

Notation: Throughout, $[n]$ denotes the set $\{1, \ldots, n\}$.

1. More Probability Practice (8 points)

1. (2 points) Explicitly calculate the probability of hitting 60 or more heads when flipping a fair coin 100 times independently, and compare this with the Chernoff bound (for both variants shown in class). Do the same for 600 or more heads in 1000 flips.

   Let $\mathbf{H}_1$ be the number of heads seen in 100 independent trials, and $\mathbf{H}_2$ be the number seen when flipping 1000 times.

   Explicit Computation:

$$\Pr[\mathbf{H}_1 \geq 60] = \sum_{i=60}^{100} \binom{100}{i} \cdot \frac{1}{2^{100}} = 0.0284$$

$$\Pr[\mathbf{H}_2 \geq 600] = \sum_{i=600}^{1000} \binom{1000}{i} \cdot \frac{1}{2^{1000}} = 1.36 \times 10^{-10}$$

   Chernoff 1: We have $\mathbb{E}[\mathbf{H}_1] = 50$ and $\mathbb{E}[\mathbf{H}_2 = 500]$ and thus apply Chernoff bound with deviations $\delta_1 = 1/5$ and $\delta_2 = 1/5$.

$$\Pr[\mathbf{H}_1 \geq 60] \leq \left(\frac{e^{0.2}}{1.2^{1.2}}\right)^{50} = 0.391$$

$$\Pr[\mathbf{H}_2 \geq 60] \leq \left(\frac{e^{0.2}}{1.2^{1.2}}\right)^{500} = 0.0000833$$

We can see that the bound is quite loose for the small sample size but not bad for the large one. It is orders of magnitude off, but still shows that the probability is very small.

Chernoff 2:

$$\Pr[\mathbf{H}_1 \geq 60] \leq 2\exp\left(\frac{-0.2^2 \cdot 50}{2.2}\right) = 0.806$$

$$\Pr[\mathbf{H}_2 \geq 600] \leq 2\exp\left(\frac{-0.2^2 \cdot 500}{2.2}\right) = 0.000225$$

2. (3 points) We would like to construct a random permutation of $[n]$, given a blackbox that outputs numbers independently and uniformly at random from $[k]$ for $k \geq n$. If we compute a function $f : [n] \to [k]$ with $f(i) \neq f(j)$ for all $i \neq j$, then this yields a permutation: simply output the numbers in $[n]$ according to the order of the $f(i)$ values. To construct such a function, do the following: iterate though each of $1, \ldots, n$ and for each, choose $f(j)$ by repeatedly obtaining numbers from the black box and setting $f(j)$ to the first number found such that $f(j) \neq f(i)$ for $i < j$.

   Prove that this approach gives a permutation of $[n]$ chosen uniformly at random from all permutations. Find the expected number of calls to the black box that are needed when $k = n$ and $k = 2n$. For the case $k = 2n$, give an upper bound (as a function of n) on the probability that the number of calls to the black box is $> 4n$.

   Let $\mathbf{T}$ be the number of call made to the blackbox. For $k = n$ this is exactly the coupon collector problem, and so the expected number of calls is $\mathbb{E}[\mathbf{T}] = \sum_{i=1}^{n} \frac{n}{i} = n \cdot H_n = O(n \log n)$. For $k = 2n$, the expected number of calls is $\mathbb{E}[\mathbf{T}] = \sum_{i=n+1}^{2n} \frac{2n}{i} = 2n \cdot (H_{2n} - H_{n+1})$. Observe that all terms in the sum at at most 2. Thus, $\mathbb{E}[\mathbf{T}] \leq 2n$.

   When $k = 2n$, every call to the black box returns a new output with probability $\geq 1/2$. Thus, to make $4n$ calls, we must have $> 3n$ that do not return a new output. I.e., the probability that we make $4n$ calls is at most the probability of flipping $3n$ tails on $4n$ fair coin flips. By a Chernoff bound with expectation $\mu = .5 * 4n = 2n$ and $\delta = 0.5$, this probability is at most:

$$\Pr[\mathbf{X} \geq 3n] \leq \left(\frac{e^{.5}}{1.5^{1.5}}\right)^{2n} = .805^n$$

   This is extremely small for large n.

3. (3 points) In practice, a *fully random hash function* that maps any input to a uniform and independently chosen output is not efficiently implementable. So, random hash functions that approximate the behavior of a fully random hash function are often used. A k-universal hash function $\mathbf{h} : U \to [n]$ is any random hash function that satisfies, for any inputs $x_1, \ldots, x_k \in U$,

$$\Pr[\mathbf{h}(x_1) = \mathbf{h}(x_2) = \ldots = \mathbf{h}(x_k)] \leq \frac{1}{n^{k-1}}.$$

   Thus is a weaker variant of a *k-wise independent hash function*. Suppose you hash n balls into n bins using a 2-universal hash function. Show that for $t = c\sqrt{n}$ for some large enough constant c, then the maximum load on any bin exceeds t with probability at most $1/10$. **Hint:** Use linearity of expectation but don't use a union bound.

Generalize this result to $k > 2$. For what value of t is the probability of the maximum load exceeding t at most $1/10$?

Let $\mathbf{C}$ be the number of pairwise collisions when using a 2-universal hash function. $\mathbb{E}[\mathbf{C}] = \frac{\binom{n}{2}}{n} \leq \frac{n^2}{2n} \leq \frac{n}{2}$. Thus, by Markov's inequality, $\Pr[\mathbf{C} \geq 5n] \leq 1/10$. If the maximum load is $\geq c\sqrt{n}$, then the number of pairwise collisions, just considering the heaviest bucket is $\binom{c\sqrt{n}}{2} \geq 5n$ for large enough c. Thus, the maximum load exceeds $c\sqrt{n}$ with probability at most $1/10$.

For general k: if the maximum load is t then there are at least $\binom{t}{k}$ k-wise collisions, just considering the maximally loaded bucket. The expected number of k-wise collisions is $\frac{\binom{n}{k}}{n^{k-1}}$, by the k-universal property. Thus, by Markov's inequality, the probability of the maximum load being $\geq t$ is upper bounded by:

$$\frac{\binom{n}{k}}{n^{k-1}\binom{t}{k}} = \frac{n(n-1)\cdots(n-k+1)}{t(t-1)\cdots(t-k+1)\cdot n^{k-1}} \leq \frac{n}{t(t-1)\cdots(t-k+1)}$$

Thus, for large enough n and large enough constant c, if $t = cn^{1/k}$ this probability will be upper bounded by $1/10$.

2. Communication Games (10 points)

1. (3 points) Consider the following communication problem: Alice and Bob both have n-bit strings a, b. If $a \neq b$, both must output the index of some position on which their inputs *do not match*. If they are given $a = b$, they can output anything they want. Describe a randomized protocol for solving this problem with probability $\geq 2/3$ and give a bound on its communication complexity in terms of bits. You may assume that players have access to a shared source of random bits.

   Use an ℓ_0 sampler – using shared random bits Alice and Bob pick the same random sketching matrix $\mathbf{A} \in \mathbb{R}^{O(\log n)\times n}$. Alice computes $\mathbf{A}a$ and sends it to Bob, who then computes $\mathbf{A}a - \mathbf{A}b$. The position of any nonzero in this vector is a position where $a(i) \neq b(i)$.

   The ℓ_0 sketch has $\log_2 n$ levels of sampling, and for each it sends 3 numbers, each requiring $O(\log n)$ bits to represent, since each is bounded in magnitude by $\text{poly}(n)$. One instance of the sketch succeeds with probability $\geq 1/8$ by the analysis shown in class. Thus, if we repeat the sketch c times for a large enough constant c, with probability $\geq 2/3$, at least one of the repetitions succeeds in outputting a non-zero element. So overall, the total communication is $O(\log^2 n)$ bits.

   Recently it was shown that any protocol requires at least $O(\log^2 n)$ bits: `https://arxiv.org/pdf/1703.08139.pdf`. So this result is tight.

2. (3 points) Prove a lower bound on the bits of communication needed for any deterministic protocol to solve the above problem. As in class, you may restrict yourself to considering protocols in which the players alternate sending 1-bit at a time.

   We know from class that Alice and Bob need to exchange $\Omega(n)$ bits to solve the equality testing problem. A protocol for this problem can be used to solve equality testing: run the protocol, which will output some index i. If $a \neq b$, then $a(i) \neq b(i)$, and the players can determine this by sending 2 additional bits. If $a = b$ then i can be arbitrary, or the protocol may not even output a valid index. In the later case, just set $i = 1$ as a default. In any case, no matter index i the players check, they will see that $a(i) = b(i)$ and so know that $a = b$. Thus, the communication complexity of this problem is $\Omega(n)$, as otherwise, it would violate the lower bound for checking equality.

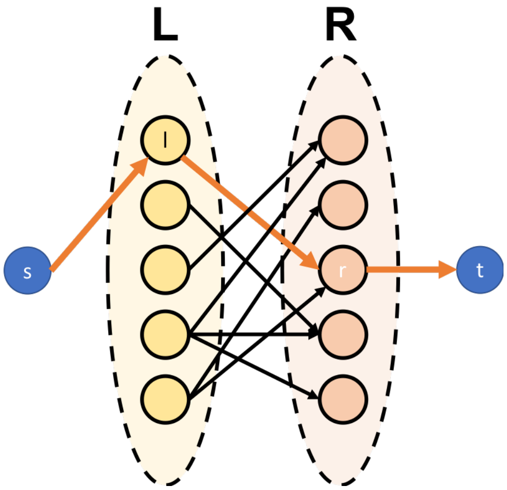
See this paper for tight deterministic lower and upper bounds for the problem: `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.42.5697&rep=rep1&type=pdf`.

3. (4 points) Consider the setting discussed in class, where n nodes of a graph know only their neighborhood and would like to send messages to a central server, such that the server can determine if the graph is connected with high probability. Consider the harder problem of directed connectivity: the graph is directed and each node only knows its outgoing edges. There are two nodes, s, t, known to all, and the central server wants to determine if there is a directed path from s to t.

   Show that if the nodes are only allowed one-way communication to the central server, then this problem requires $\Omega(n^2)$ total bits of communication to solve (as opposed to $O(n \cdot \log^c n)$ for undirected connectivity.

   Hint: Try to use an indistinguishability argument like we did for the communication complexity of equality testing in class. It suffices to consider directed acyclic graphs, and you may assume for simplicity that the protocol is deterministic – there is not a significant gap between randomized and deterministic algorithms here.

   Consider the graph where you have two sets of $(n-1)/2$ nodes, L and R. All edges from L lead to R. Let s have an outgoing edge to an arbitrary node in $l \in L$ and t have an incoming edge from an arbitrary node $r \in R$. Observe that there is a directed path from s to t if and only if (l, r) is in the edge est. See below for an illustration:



   There are $2^{\binom{(n-1)/2}{2}} = 2^{\Omega(n^2)}$ possible edge patterns between L and R. Assume for the sake of contraction that the protocol uses $o(n^2)$ bits. This means the nodes in L send $o(n^2)$ bits. Thus, there are $2^{o(n^2)}$ possible transcripts of what they send, and so there must be at least two different configurations of the edges between L and R such that they send exactly the same bits. Observe that since the other nodes (s,t, and those in R) do not see any of the edges between L and R they also send the same bits in both configurations. However, if these configurations differ on edge (l, r), then in the case that s connects to l and r connects to t, one of the graphs will have a path from s to t while the other won't. Since all nodes send the same messages in both cases, the central server must be wrong in at least one of them.

3. Sketching for Minimum Spanning Tree (6 points)

The ℓ_0 sampling algorithm for graph connectivity described in class in fact does not just determine connectivity, but, if the graph is connected, outputs a spanning tree. Consider the more difficult problem of outputting a *minimum weight spanning tree*. Show how to solve this problem with high probability (i.e., $\geq 1-1/n^c$ for some constant c), using just $O(n\log^c n)$ total bits of communication for some constant c. You may use several rounds of interaction with the central server, however any messages sent by the central server back to the nodes must be included in your accounting of the communication complexity. You may assume that edges have positive integer weights bounded by n^c for some constant c. A weight of '0' indicates that the edge is not present in the graph.

Hint: Implement Boruvka's algorithm for minimum spanning tree – instead of picking an arbitrary outgoing edge from each connected component, pick the minimum weight outgoing edge. The challenge is to figure out how each node can identify the minimum weight outgoing edge from their connected component after they start being merged together.

Each server sends just their minimum weight outgoing edge in the first round. The central server uses these to initially merge components. Each subsequent round of Boruvka's algorithm (of which there are $\leq \log_2 n$) is simulated over $O(\log n)$ rounds of communication. Nodes initially send ℓ_0 sampling sketches of their whole neighborhoods. The central server then picks an outgoing edge out of each connected component. With at least $1/2$ probability, this edge has weight $\leq 1/2$ of the outgoing edges from that component. The central server sends this edge weight back to each node in the connected component, and they recompute ℓ_0 sampling sketches, but excluding any edges with weight at least as large as the one that was just selected. Thus with probability $\geq 1/2$, at least $1/2$ of the highest weight edges leaving the component are excluded. This process repeats, and after $O(\log n)$ rounds, with high probability, the edge recovered by the central server will in fact be the minimum weight outgoing edge from the connected component, since all others will have been removed. The server will know this since, once it selects the minimum weight outgoing edge, no edges will be sent in the next round, and so all the ℓ_0 sampling sketches will equal 0. This edge is then added to the MST, components are contracted, and the algorithm moves to the next round.

To formally show that $O(\log n)$ rounds of interaction are needed to identity the minimum weight edge with high probability: Consider using t rounds of interaction. Let $\mathbf{X}$ be the number of rounds in which an edge is selected whose weight was larger than $1/2$ of the remaining edges. $\mathbb{E}[\mathbf{X}] \leq t/2$. If the minimum weight edge is not identified after t rounds, it must be that $\mathbf{X} \geq t - \log_2 n$. If $t = c\log_2 n$ for a large enough constant c, we can apply a Chernoff bound:

$$\begin{aligned}\Pr[\mathbf{X} \geq t - \log_2 n] = \Pr[\mathbf{X} \geq (c-1)\log_2 n] &= \Pr[\mathbf{X} \geq \frac{2(c-1)}{c}\mathbb{E}[\mathbf{X}]] \\ &\leq \left(\frac{e^{2(c-1)/c}}{(1+(c-1)/c)^{1+(c-1)/c}}\right)^{c\log_2 n/2} \leq \frac{1}{n^{c'}},\end{aligned}$$

where c' is a constant depending on c.