# 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.

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 $> 4n$.

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 $1/10$?

## 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.

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.

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.

## 3. Sketching for Minimum Spanning Tree (6 points)

The $\ell_0$ 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.