Let \( G \) be a (directed) graph, \( s \) a source, and \( t \) a sink. Two players play the following “intrusion” game. Player 1 picks a path \( P \) from \( s \) to \( t \) (possibly randomly), while player 2 picks a set \( S \) of \( r \) edges of \( G \). Think of player 1 taking \( P \) to get from \( s \) to \( t \), while player 2 places checkpoints (or patrols) on the edges in \( S \). If \( P \cap S = \emptyset \), then player 1 manages to get to \( t \) undetected, and wins. If \( P \cap S \neq \emptyset \), then player 2 catches player 1, and wins. Player 1 wants to minimize the probability of being caught, while player 2 wants to maximize the probability of catching player 1. Let \( M \) be the number of edges in a minimum \( s \)-\( t \) cut.

(a) Give a (possibly randomized) strategy for player 1 to be caught with probability at most \( \min(r/M, 1) \).

(b) Prove that no strategy (randomized or deterministic) can have a probability of being caught strictly less than \( \min(r/M, 1) \) against a player 2 who plays perfectly.

Open Problem: Suppose that there are two sinks \( t_1, t_2 \). If player 1 manages to get to \( t_1 \) undetected, he wins one point. If he gets to \( t_2 \) undetected, he wins two points. If he is caught, player 2 wins. (It doesn’t matter if we call this 0 or 1 points for player 2.) Give an algorithm that computes an optimal randomized strategy for player 1 and/or player 2 (assuming that that player must go first).

With Halloween rolling around, we have to start worrying about zombie attacks. Here is how zombie attacks work. You start with just one zombie in a population of \( n \) people. In each round, each zombie independently picks some moving thing uniformly at random and bites it. If that moving thing was another zombie, nothing happens. If it was a human, he/she will now also turn into a zombie for the next round. Note that multiple zombies might choose to bite the same zombie or human.

We are interested in how many rounds it will take until there are no more humans left. Show that with high probability, this happens after no more than \( O(\log n) \) rounds.

Suppose that we generate a string \( x \) of length \( n \) over an alphabet \( \Sigma \) (such as a genome over \( \Sigma = \{a, c, g, t\} \)) uniformly at random by generating each position independently and uniformly at random. We are now interested in how many occurrences of a pattern \( p \) there are in \( x \). \( p \) is also a string (of length \( k \leq n \)) over \( \Sigma \).

(a) Derive the expected number of occurrences of \( p \) in \( x \).

(b) Show that the actual number of occurrences is sharply concentrated around the expectation. How sharp a bound can you obtain?

Here is a variant of the well-known Hitting Set problem. There is a universe \( U \) of \( n \) elements, and you have \( m \) possible sets \( S_1, \ldots, S_m \subseteq U \), with associated integer “coverage requirements” \( r_1, \ldots, r_m \geq 1 \). To keep things clean, let’s assume that \( m \) is polynomial in \( n \) always.

You want to select a subset \( T \subseteq U \) of elements, ensuring that you hit each set \( S_j \) at least \( r_j \) times, i.e., that \( |T \cap S_j| \geq r_j \) for all \( j \). Subject to that, you want to minimize the size of \( T \).

Phrase this question as an integer linear program, and consider its fractional relaxation. Give a suitable rounding algorithm of this linear program, and prove that it gives an \( O(\log n) \) approximation algorithm.