1. (**Isolated vertices in $\mathcal{G}_{n,p}$.**) In this problem, we will see that the value $p = \frac{\ln n}{n}$ is a "threshold" for the property that a random graph in the $\mathcal{G}_{n,p}$ model has an isolated vertex, i.e. a vertex with degree 0. That is, we will prove that

$$\Pr[G \text{ has an isolated vertex}] \overset{n\to\infty}{\longrightarrow} \begin{cases} 0 & \text{if } p = \omega(\frac{\ln n}{n}) \\ 1 & \text{if } p = o(\frac{\ln n}{n}) \end{cases}$$

(a) Let the r.v. X denote the number of isolated vertices in G. Write down the expectation of X as a function of n and p.

(b) Show that $\mathrm{E}[X] \to 0$ for $p = \omega(\frac{\ln n}{n})$ and that $\mathrm{E}[X] \to \infty$ for $p = o(\frac{\ln n}{n})$.
[HINT: Write $p = a \cdot \frac{\ln n}{n}$ and observe that $\mathrm{E}[X] \approx n^{1-a}$.]

(c) Deduce from part (b) that $\Pr[G \text{ has an isolated vertex}] \to 0$ for $p = \omega(\frac{\ln n}{n})$.

(d) Show that $\mathrm{Var}[X] = n(1-p)^{n-1} + n(1-p)^{2n-3}(np-1)$.

(e) Deduce from parts (b) and (d) that $\Pr[G \text{ has an isolated vertex}] \to 1$ for $p = o(\frac{\ln n}{n})$.
[HINT: First show that $\Pr[X = 0] \le \frac{1}{\mathrm{E}[X]} + \frac{p}{1-p}$.]

2. (**Tournament Rankings.**) MU Ex 6.9.

[HINT: For part (b), you will need to first use a Chernoff bound and then a union bound.]

3. (**DNF Approximate Counting.**) A fundamental problem that arises in many applications is to compute the size of the *union* of a collection of sets. The setting is the following. We are given m sets $S_1, \ldots, S_m$ over a very large universe U. The operations we can perform on the sets are the following:

- $\mathrm{size}(S_i)$: returns the number of elements in S_i;
- $\mathrm{select}(S_i)$: returns an element of S_i chosen uniformly at random;
- $\mathrm{lowest}(x)$: for $x \in U$, returns the smallest index i for which $x \in S_i$.

Let $S = \cup_{i=1}^m S_i$ be the union of the sets S_i. In this problem, we will develop a very efficient (polynomial in m) algorithm for estimating the size $|S|$ up to *multiplicative* factors (see part (f) for a formal definition).

(a) Let's first see a natural example where such a set system arises. Suppose ϕ is a boolean formula in *disjunctive normal form* (DNF), i.e. it is the OR of ANDs of literals. Let U be the set of all possible assignments to the variables of ϕ (i.e., $|U| = 2^n$ where n is the number of variables), and for each clause $1 \le i \le m$, let S_i be the set of assignments that satisfy clause i. Then the union $S = \cup_{i=1}^m S_i$ is exactly the set of satisfying assignments of ϕ, and our problem is to count them.[1] Argue that all of the above operations can be efficiently implemented for this set system.

[1] Note that deciding if ϕ is satisable (i.e., has at least one satisfying assignment) is trivial for a DNF formula, unlike for a CNF formula where it is NP-complete. However, when it comes to counting satisfying assignments, it turns out that the problem is NP-hard even for DNF formulas! Thus we cannot hope to find a polynomial time algorithm that solves this problem exactly. Thus the approximation algorithm that we develop in this question is essentially the best one can hope for.

(b) Now let's consider a naive sampling algorithm. Assume that we are able to pick an element of U uniformly at random, and that we know the size of U. Consider the algorithm that picks t elements of U independently and uniformly at random (with replacement), and outputs the value $q|U|$, where q is the proportion of the t sampled elements that belong to S. For the DNF example in part (a), explain as precisely as you can why this is not a good algorithm.

(c) Consider now the following algorithm, which is again based on random sampling but in a more sophiscated way:

- choose a random set S_i with probability $\frac{\text{size}(S_i)}{\sum_j (S_j)}$
- $x = \text{select}(S_i)$
- if $\text{lowest}(x) = i$ then output 1 else output 0 Show that this algorithm outputs 1 with probability exactly $p = \frac{|S|}{\sum_j |S_j|}$. [HINT: Show that the effect of the first two lines of the algorithm is to select a random element of the set of pairs $\{(x, S_i) : x \in S_i\}$.]

(d) Show that $p \geq \frac{1}{m}$.

(e) Now suppose we run the above algorithm t times and obtain the sequence of outputs $X_1, \ldots, X_t$. We define $X = \frac{1}{t}\sum_{i=1}^{t} X_i$. Use the Chernoff bound to obtain a value for t (as a function of m, δ, ϵ) that ensures that

$$\Pr\big[|X - p| \geq \epsilon p\big] \leq \delta.$$

[HINT: You will need to use the bound from part (d) here.]

(f) The final output for our algorithm will be $Y = (\sum_j |S_j|) \cdot X$, where X is defined as in part (e). Using part (e), show that this final algorithm has the following properties: it runs in time $O(m^2\epsilon^{-2}\log(\delta^{-1}))$ (assuming that each of the set operations listed above can be performed in constant time), and output a value that is in the range $[(1-\epsilon)|S|, (1+\epsilon)|S|]$ with probability at least $1 - \delta$.