1. (Credit: HW problem from a class by Avrim Blum and Anupam Gupta at CMU) Given a graph $G = (V, E)$ with $|V| = n$ and $|E| = m$ as usual, we aim here to efficiently find long simple paths (those in which no repeated vertex exists) in $G$. The length of a path as usual denotes the number of edges in it.

- (Algorithm 1: Very easy.) Show how to find a simple path of length $k$ (if one exists) in time $O(n\Delta^k)$, where $\Delta$ as usual denotes the maximum degree. (We will try to do better next.) (5 points)

- (Easy) If $G$ is a DAG (a directed acyclic graph), show that you can deterministically find the longest path in $G$ in $O(m + n)$ time. (7 points)

- (Algorithm 2:) Consider running the following algorithm $n$ times, and outputing the longest path found in these $n$ trials:

  Take a random permutation of the vertices, and direct each edge from the lower endpoint to the higher endpoint to create a DAG $H$. Find a longest path in $H$.

Show that there is some constant $c > 0$ such that for $k \leq c\log n / \log \log n$, Algorithm 2 will find a simple path of length $k$ (if one exists) with probability at least $1/2$. (10 points)

- Now consider a slight extension of the last-seen idea. Suppose we have the set $K = \{1, 2, \ldots, k+1\}$ of labels, and we label each vertex with some element of $K$. A path is called polychromatic if all its vertices get different labels (note that such a path can be of length at most $k$).

  - Given a labeling, show how to find a polychromatic path of length $k$ (if one exists) in time that is $O(poly(n) \cdot 2^k)$. Note that this run-time is $poly(n)$ if $k \leq O(\log n)$. (9 points)

  - (Algorithm 3:) Consider running the following algorithm $n$ times, and outputing the longest path found in these $n$ trials:

    Take a random labeling of the vertices using $K$, and output a polychromatic path of length $k$, if one exists.

Show that there is some constant $c > 0$ such that for $k \leq c\log n$, Algorithm 3 will find a simple path of length $k$ (if one exists) with probability at least $1/2$. (9 points)

2. We are given a bipartite graph $G = (U_1, U_2, E)$ with $n$ vertices in total; each vertex $v$ has a set $S(v)$ of labels given to it. Prove that there is a constant $c > 0$ such that if $|S(v)| \geq c\log_2 n$ for each vertex $v$, then there exists a choice of label $\ell(v) \in S(v)$ for each vertex $v$, so that no two adjacent vertices get the same label. (Hint: Let $S$ be the union of all the sets $S(v)$. Start with a random function $f : S \to \{1, 2\}$, and use the fact that $G = (U_1, U_2, E)$ is bipartite.) (15 points)

3. Let $X_1, X_2, \ldots, X_n \in \{0, 1\}$ be random bits that are not necessarily independent; let $X = \sum_i X_i$. Prove that for any pair of integers $a, k$ with $1 \leq k \leq a \leq n$,

$$\Pr[X \geq a] \leq \frac{\sum_{1 \leq i_1 < i_2 < \cdots < i_k \leq n} E[X_{i_1} X_{i_2} \cdots X_{i_k}]}{{n \choose k}}.$$  

As an example, if $n = 4$ and $k = 2$, then the numerator of the r.h.s. is $E[X_1 X_2] + E[X_1 X_3] + E[X_1 X_4] + E[X_2 X_3] + E[X_2 X_4] + E[X_3 X_4]$. (10 points)
4. In both instances below, \( n \) and \( d \) will denote the number of vertices, and average degree, respectively, of a given undirected graph.

It is known that for some constant \( c_0 > 0 \), every graph with no triangles has an independent set of size at least \( c_0 \cdot (n/d) \cdot \log d \). Use this to prove that there is a constant \( c_1 > 0 \) (which depends only on \( c_0 \)) such that the following holds: if graph \( G \) has at most \( \delta d^2 n \) triangles (where \( \delta \) is an arbitrary parameter such that \( 1/\sqrt{d} \leq \delta \leq 1 \)), then \( G \) an independent set of size at least \( c_1 \cdot (n/d) \cdot \log(1/\delta) \). (Hint: Use sampling and alteration.) (20 points)

Remarks. A triangle is just a cycle of length 3. Recall that we showed, using random permutations and convexity, that for any graph \( G \) (with \( n \) vertices and average degree \( d \)) has an independent set of size at least \( n/(d+1) \); this is called Turán’s Theorem. What the “\( c_0 \cdot (n/d) \cdot \log d \)” above says that once we have the triangle-freeness condition, we gain a multiplicative factor of \( \Theta(\log d) \). You can always assume that \( d \) is “large enough”, say \( d \geq 4 \); if \( d < 4 \), we can just use Turán’s Theorem and take \( c_0 > 0 \) small enough. The above condition \( \delta \geq 1/\sqrt{d} \) is not strictly necessary, but makes your calculations a bit easier.