Due: Wednesday February 11th, in class.

Question 1: Super-Sparse Sampling Works
Prove the claim about super-sparse sampling from Lecture 8.

Claim 1. Let \( y \) be a fixed vector in \( \mathbb{R}^d \) with \( \|y\|_2 = 1 \) and
\[
\|y\|_{\infty} \leq \lambda = \sqrt{\frac{2 \ln(4d/\delta)}{d}}.
\]
Let \( S \) be a \( t \times d \) super-sparse sampling matrix with \( t = 2 \ln(4d/\delta)^2 \ln(4/\delta)/\epsilon^2 \). Then
\[
\Pr \left[ \|Sy\|_2^2 \not\in (1 - \epsilon, 1 + \epsilon) \right] \leq \delta/2.
\]
Hints:
- \( \|Sy\|_2^2 = \sum_i (Sy)_i^2 \), and these summands are independent.
- The expectation was already analyzed in Lecture 8.
- The Generalized Hoeffding bound from Lecture 8 is probably more convenient than the Chernoff bound from Lecture 3.

Question 2: Johnson-Lindenstrauss Implementation
Please implement the Johnson-Lindenstrauss dimensionality reduction algorithm in your favorite programming language (Matlab, Python, etc).

Try applying the algorithm to a few simple data sets, such as randomly distributed data, random clusters of data, highly structured data, or even some real-world data. Some possible parameter settings might be \( n \approx 10000, d \approx 4000, \epsilon \approx 0.25 \).

In Lecture 7, the embedding dimension was chosen to be \( t = (4/3) \ln(n^3)/\epsilon^2 = 4 \ln(n)/\epsilon^2 \). Is that too conservative? If your low-dimensional space has dimension \( c \ln(n)/\epsilon^2 \), what value would you suggest for the constant \( c \) in order to preserve all pairwise distances up to \( 1 \pm \epsilon \)?

Let us say that the “distortion” of a vector is the its norm in original space divided by its norm in the low-dimensional space. If we look at all pairs of points in the data set, how are their distortions distributed? Do many of them have distortion close to \( 1 - \epsilon \) or \( 1 + \epsilon \)?

Question 3: Minimum Cut

(a): Generalizing on the notion of a cut-set, we define an \( k \)-way cut-set in an undirected graph as a set of edges whose removal breaks the graph into \( k \) or more connected components. Explain how the randomized min-cut algorithm can be used to find minimum \( k \)-way cut sets. Bound the probability that it succeeds in one iteration and bound the total running time for it to have success probability at least \( 1 - 1/n \) where \( n \) is the number of vertices in the graph.

(b): Given an undirected graph \( G \) with minimum-cut size \( c \), prove that \( G \) has at most \( O(n^{2\alpha}) \) cuts with at most \( \alpha c \) edges.