1. (40 points) **(Fun with Metrics)** In this exercise, you will learn how to use metric embeddings to design algorithms.

   (a) (10 points) Show how to solve the k-median problem on line metrics optimally. That is, given n clients C and m choices for placing the centers F, and a line metric d_L on $C \cup F$, find a subset $S \subseteq F$ such that $|S| = k$ and the $cost(C, S)$ is minimized. Recall that $cost(C, S) = \sum_{j \in C} d_L(j, S)$, and $d_L(j, S) = \min_{i \in S} d_L(j, i)$.

   > **Solution:**
   >
   > *Proof.* $1 + 1 = 2$. $\qquad\square$

   (b) (10 points) Now assume the following (fake) theorem:
   Theorem 1. *Given any metric (X, d) where X is a set of n points, and $d(\cdot)$ is a general metric distance function, we can efficiently find an embedding into a line metric d_L such that for all $u \neq v$, $d(u, v) \leq d_L(u, v) \leq O(\log n)d(u, v)$.*

   Using the fake theorem above, devise a $O(\log n)$-approximation to k-median on general metrics. (Hint: consider the modified instance $\mathcal{I}' = (C \cup F, d_L)$ and solve k-median on d_L, and output the same solution for the original instance.)

   > **Solution:**
   >
   > *Proof.* $1 + 1 = 2$. $\qquad\square$

   (c) (10 points) In reality the fake theorem is not true as stated above. What is true is the following theorem:
   Theorem 2. *Given any metric (X, d) where X is a set of n points, and $d(\cdot)$ is a distance function, we can efficiently find an embedding into a distribution $\mathcal{D}$ over different line metrics $d_{L_1}, d_{L_2}, \ldots, d_{L_k}$ such that for all $u \neq v$, $d(u, v) \leq \mathbb{E}_{d_L \sim \mathcal{D}}[d_L(u, v)] \leq O(\log n)d(u, v)$.*

   Now, can you extend the same reasoning to devise good approximation algorithms for k-median? If not, what hurdles do you face? For a hint, see the part below.

   > **Solution:**
   >
   > *Proof.* $1 + 1 = 2$. $\qquad\square$

   (d) (10 points) What happens if you further assume in the theorem (in reality this assumption is not valid, but let us just assume it for the sake of this question) that for all line metrics d_{L_i} in the support of $\mathcal{D}$ and all $u \neq v$, we have $d_{L_i}(u, v) \geq d(u, v)$, and $\mathbb{E}_{d_L \sim \mathcal{D}}[d_L(u, v)] \leq O(\log n)d(u, v)$. Such embeddings are called embeddings into a distribution of **dominating lines** (for any pair $u \neq v$, the embedding function *always increases* the distances, but *on expectation*, does not increase by more than a $O(\log n)$ factor). Now does it help you devise approximation algorithms?

2. (40 points) **(Algorithms for Warehouse Placement)** We now look at a modification of the k-median problem, which we call warehouse placement. Here, we're given a metric space with n retail outlets (or clients) denoted by set C, and m possible locations F, and a metric d over $C \cup F$. We want to place k warehouses at a subset $S \subseteq F$ such that $\max_{j \in C} d(j, S)$ is minimized. Notice that if the max is replaced by $\sum$, then we will get back the k-median problem.

   (a) (10 points) Show that the problem does not admit any $(3 - \epsilon)$-approximation algorithm if $P \neq NP$ (Hint: reduce from Max-k-Coverage).

   > **Solution:**
   >
   > *Proof.* $1 + 1 = 2$. □

   Now analyze the following algorithm for this problem: suppose somehow we know the value of the optimal solution C^*. Then, build a graph over the clients C with the following edges: place an edge between $j_1 \in C$ and $j_2 \in C$ if and only if $d(j_1, j_2) \leq 2C^*$.

   (b) (10 points) Show that the size of any maximal independent set in this graph is at most k.

   > **Solution:**
   >
   > *Proof.* $1 + 1 = 2$. □

   (c) (10 points) Show how to use some maximal independent set found to recover a set $S \subseteq F$ of k locations with a good approximation ratio for the original problem. What's the best approximation factor you can get?

   > **Solution:**
   >
   > *Proof.* $1 + 1 = 2$. □

   (d) (10 points) Show how to dispense the assumption of knowing C^* by trying all distance values and enumerating.

   > **Solution:**
   >
   > *Proof.* $1 + 1 = 2$. □