Note: In all these problems: if we ask you to show an approximation ratio of ρ, you should tell us about the best approximation ratio you did prove, be it better or worse than ρ. Also, please refrain from consulting any external sources (papers, lecture notes, books, surveys) in attempting to solve these problems.

1. **(Directed Spanning Trees via the Local Ratio Method)** In this problem, we are going to use the Local Ratio method to devise an *exact* algorithm for the minimum-cost directed spanning tree problem. In this problem, given a *directed* graph $G = (V, A)$ with nonnegative costs on the arcs $c : A \to \mathbb{R}_{\geq 0}$ and a root node r, the objective is to find a *directed spanning tree* $T = (V, A')$ rooted at r; in such a tree, there is a directed path *from* every node to r, and the tree has exactly $|V| - 1$ arcs. (While we will not say "directed" repeatedly, all spanning trees in this problem are directed.)

   As always, the first step is to decide the cost function $\bar{c}$: for this, we consider the subgraph $Z = (V, A_Z)$ consisting of all the zero-cost edges in G. (I.e, $A_Z = \{e \in A \mid c_e = 0\}$.)

   - If each vertex in V can reach the root in Z, then there is a zero cost solution, and we are done.
   - Suppose there is some strongly connected component C of Z that has no zero-cost edges coming out of it. (Note that C is may consist of a single node.) In that case, we define $\alpha = \min_{e \in \partial^+(C)} c_e$, and we set $\bar{c}_e = \alpha$ for $e \in \partial^+(C)$ and zero otherwise. (Recall that $\partial^+(S)$ is the set of arcs going out of $S \subseteq V$.)
     To get a reduced graph G', we contract the strongly connected component C into a new supernode v_C. We now set $c'_e = c_e - \bar{c}_e$ for all arcs, and find the optimal spanning tree on (G', c'). While returning from the induction in this case, we have an optimal spanning tree T' of the graph G': suppose the arc (v_C, u) in this tree out of the supernode v_C corresponds to the arc (v, u) in the original graph G, with $v \in C$. We now "unshrink" the supernode v_C, and add to the tree T' a spanning tree T_C of zero-cost edges within the strongly connected component C rooted at v obtain a spanning tree of the whole unshrunk graph.

   Your task is now to use the above idea to obtain an exact algorithm with a proof.

   (a) Write out the above outline as a recursive procedure for finding a spanning tree rooted at r. (You may want to try your algorithm on small examples to make sure it works.)
   (b) Argue that the set of edges output is a feasible spanning tree by induction. Show carefully the inductive step of obtaining a spanning tree rooted at any node from a strongly connected subgraph.
   (c) Argue that the cost of *any* minimum spanning tree on graph $(G, \bar{c})$ is at least α.
   (d) Using the previous part and induction, prove that the cost of the tree output is the minimum possible.
   (e) Analyze the running time of the above algorithm and prove the most efficient bound you can.

2. **(Approximation via Randomized Simplification)** In this part we study the application of probabilistic embeddings of a metric into spanning trees to approximation algorithms.

   Given a set X of *points*, a *distance function* on X is a map $d : X \times X \to \mathbb{R}_+$ that is symmetric, and satisfies $d(i, i) = 0$ for all $i \in X$. The distance is said to be a *metric* if the *triangle inequality* holds, i.e.,

   $$d(i, j) \leq d(i, k) + d(k, j) \qquad \forall i, j, k \in X.$$

   We informally think of metrics as a complete graph on X with the edge $\{i, j\}$ having a length $d(i, j)$.

   We say that the metric (X, d) *α-probabilistically embeds into a probability distribution $\mathcal{D}$ of trees* if

- Each tree $T = (V_T, E_T)$ in the support of the distribution $\mathcal{D}$ has $V_T = X$.
- The distances in each such tree T *dominate* those in d; i.e., $d_T(x, y) \geq d(x, y)$ for all $x, y \in X$.
- Given a pair of vertices $x, y \in X$, the expected distance is "not too much larger" than $d(x, y)$: formally
$$\mathbf{E}[\, d_T(x, y)\,] \leq \alpha\, d(x, y).$$

The following result due to Fakcharoenphol, Rao and Talwar (2003) gives the best-possible results of type. (We will see a proof later in class.)

Theorem 1 *Any n-point metric α-probabilistically embeds into a distribution of trees $\mathcal{D}_{FRT}$ with $\alpha = O(\log n)$; furthermore, samples from this distribution can be generated in polynomial time.*

Our goal is to use this result to design approximation algorithms for problems defined on metrics by first designing exact or near-optimal solutions to the same problem on trees, and then using the above result to translate such an algoruthm to arbitrary metrics with a loss of $\alpha = O(\log n)$ in the performance ratio.

(a) Consider the K-median problem that we discussed in class: given an n-point metric (X, d), and a number K, find a set C of size K that minimizes $\Phi(C) = \sum_{x \in X} d(x, C)$. Suppose you are given a tree T with nonnegative edge lengths as your input to the K-median problem, show how to solve the k-median problem in polynomial time for this class of input graphs. (Hint: Use Dynamic Programming).

(b) To solve the K-median problem on a general metric, we apply Theorem 1 to the input metric to derive a probability distribution over tree metrics. We then sample a tree according to this distribution and apply your polynomial time algorithm from the previous part to it. We then use the same set of K nodes as the solution to the K-median problem on the original metric.

   i. Show that an optimal K-median solution C^* in the original metric gives a solution whose expected cost is at most $\alpha \cdot \Phi(C^*)$. Thus the optimal solution in the sampled tree has expected cost no more than this.

   ii. Show how *any* K-median solution of cost D in the sampled tree T has cost no more than D in the original metric.

   Mention how these two claims imply an α-approximation for the K-median problem on any metric.

(c) Look at your analysis in the previous part and outline for what kinds of problems on metrics does this method apply. E.g., does it work for the TSP o a metric? How about the K-center problem? Why or why not?