1. Describe and analyze an algorithm that finds a *maximum* spanning tree of a graph, that is, the spanning tree with largest total weight.

2. During your CS 473 final exam, you are given a specific undirected graph $G = (V, E)$ with non-negative edge weights, and you are asked to compute both the minimum spanning tree of $G$ and the tree of shortest paths in $G$ rooted at some fixed source vertex $s$.

   Two and a half hours into the exam, Jeff announces that there was a mistake in the exam; every edge weight should be increased by 1. Well, that's just great. Now what?!

   (a) Do you need to recompute the minimum spanning tree? Either prove that increasing all edge weights by 1 cannot change the minimum spanning tree, or give an example where the minimum spanning tree changes.

   (b) Do you need to recompute the shortest path tree rooted at $s$? Either prove that increasing all edge weights by 1 cannot change the shortest path tree, or give an example where the shortest path tree changes.

3. After graduating you accept a job with Aerophobes-Я-Us, the leading traveling agency for people who hate to fly. Your job is to build a system to help customers plan airplane trips from one city to another. All of your customers are afraid of flying (and by extension, airports), so any trip you plan needs to be as short as possible. You know all the departure and arrival times of every flight on the planet.

   Suppose one of your customers wants to fly from city $X$ to city $Y$. Describe an algorithm to find a sequence of flights that minimizes the *total time in transit*—the length of time from the initial departure to the final arrival, including time at intermediate airports waiting for connecting flights. *[Hint: Modify the input data and apply Dijkstra's algorithm.]*