1. Shortest Bottleneck Path

   Let $\mathcal{G}$ be an undirected graph with distinct nonnegative edge lengths. The *bottleneck distance* of a path in $\mathcal{G}$ is the length of the longest edge in the path.

   Prove that the minimum spanning tree of $\mathcal{G}$ contains the shortest bottleneck path between every pair of points.

2. Fixing a Minimum Spanning Tree

   Let $\mathcal{G}$ be an undirected graph with distinct edge weights. Let T be a minimum spanning tree in $\mathcal{G}$.

   Suppose the weight of a single edge e is altered. e may or may not be in T. Describe an $O(m+n)$ time algorithm that computes the MST of $\mathcal{G}$ with the updated weight.

3. Random Cuts

   Let G be a graph. A *cut* of G is a partition (A, B) of the vertices (i.e., A and B are subsets of $V(G)$, $A \cap B = \emptyset$, and $A \cup B = V(G)$. The *size* of a cut (A, B) is the number of edges $(u, v) \in E(G)$ such that $u \in A$ and $v \in B$.

   Prove that the expected size of a random cut is $|E(G)|/2$. That is, suppose that each vertex is placed in A with probability $1/2$ and in B with probability $1/2$, and let X denote the size of the resulting cut. Prove that $\mathbb{E}[X] = |E(G)|/2$.

4. Bůruvka's Algorithm

   Bůruvka's algorithm computes the MST of a graph G as follows. At the beginning of the algorithm, we have a forest where each vertex in $V(G)$ is its own connected component. At each iteration, the algorithm goes through every connected component in the graph and picks the cheapest edge adjacent to it, adding it to the forest.

   (a) Show that Bůruvka's algorithm decreases the number of vertices by a factor of two at each iteration.

   (b) Conclude that Bůruvka's algorithm takes $O(m \log n)$ time in the worst case.

5. Nuts and Bolts

   Suppose we are given n nuts and n bolts of different sizes. Each nut matches exactly one bolt and vice versa. The nuts and bolts are all almost exactly the same size, so we can't tell whether one bolt is smaller than the other, or if one nut is bigger than the other. If we try to match a nut with a bolt, it is either too big, too small, or an exact match. Describe an algorithm that matches the all the nuts and bolts in $O(n \log n)$ time in expectation.