1. In lecture we define the length of a path to be the sum of the lengths of its edges. Define the *bottleneck* of a path to be the maximum length of one of its edges. A *mininum-bottleneck path* between two vertices s and t is a path with bottleneck no larger than that of any other s-t path. Show how to modify Dijkstra's algorithm to compute a minimum-bottleneck path between two given vertices. The running time should be O(mlog⁡n), as in lecture.
2. We can do better. Suppose now that the graph is undirected. Give a linear-time (O(m)) algorithm to compute a minimum-bottleneck path between two given vertices.
3. What if the graph is directed? Can you compute a minimum-bottleneck path between two given vertices faster than O(mlog⁡n)?