Exercise 16.6. Let $G = (V, E)$ be an undirected graph with distinct nonnegative edge weights $w : E \to \mathbb{R}$. For a spanning tree T, we say that the *bottleneck weight of* T is the maximum weight edge in T, $\max_{e \in T} w(e)$.

Exercise 16.6.1. Prove that the MST is also a minimum bottleneck weight spanning tree of G.

Solution. Let α be the MST of G. Assume ad absurdum that there exists an minimum bottleneck weight spanning tree (MBWST) β of G such that $\alpha \neq \beta$. Then by definition of bottleneck weight, we have that

$$\max_{e \in \alpha}\{w(e)\} > \max_{e \in \beta}\{w(e)\}.$$

That is to say, there exists some edge $e = (v_1, v_2) \in \alpha$ such that $w(e)$ is greater than $w(e')$ for every $e' \in \beta$. Obviously $e \notin \beta$, but by definition of spanning tree β spans e. Since α does not contain any cycles by definition of tree, WLOG there exists some $f = (v_1, v_3) \in \beta$ such that $f \notin \alpha$. Then we have that

$$w(e) > w(f).$$

By the spanning property of α, we know there exists an edge $(v_3, v_k) \in \alpha$ for some arbitrary vertex v_k. Thus by removing e and adding f to α, we can preserve the spanning property of α while reducing its total weight, contradicting that α is the MST of G. Thus the minimum spanning tree of G must also be a minimum bottleneck spanning tree of G. □

Exercise 16.6.2. Design and analyze a $O(m+n)$-time algorithm for computing a minimum bottleneck weight spanning tree of G. (This is faster than any of our algorithms for MST.)[4]

[4]Here's step 1: compute the median edge weight in $O(m)$ time.

Solution. Our algorithm is as follows:

```
MBWST($G = (V, E)$):
1. Compute the median edge weight $w_{\text{med}}$ using median of medians    // $O(m)$
2. Partition $E$ into $E_{\leq} = \{e \in E : w(e) \leq w_{\text{med}}\}$ and $E_{>} = \{e \in E : w(e) > w_{\text{med}}\}$
3. Run BFS to check if $G_{\leq} = (V, E_{\leq})$    // $O(m+n)$
```

If $G_{\leq}$ is connected, then there is a spanning tree all of whose edges have weight at most w_{med}. In particular, the MBWST has bottleneck weight at most w_{med}. We can thus restrict our search to the edges in $E_{\leq}$.

Else, any spanning tree must use at least one edge from $E_{>}$. In this case, we can contract each connected component of $G_{\leq}$ into a single supervertex. Form the new graph G' on these supervertices where each edge corresponds to an edge in $E_{>}$ (with the same weight). Notice that a spanning tree of G corresponds to a spanning tree of G' once we add back any spanning trees for the contracted components (which can be computed via BFS). We then continue our search on G' with the new edge set.

Runtime Complexity. Each iteration uses $O(m+n)$ time for the median selection and connectivity test. Further, in each case at least about half of the edges are discarded (or G is contracted such that the total number of edges in the G' decreases). This gives a recurrence of the form

$$T(m) = T\left(\frac{m}{2}\right) + O(m+n)$$

which solves to $O(m+n)$. ■

Correctness. By recursively reducing the set of candidate edges, we eventually obtain a subgraph that is connected and such that the highest weight in the MBWST is the smallest possible threshold guaranteeing connectivity. Finally, once the threshold is determined, a spanning tree taken from the subgraph $G_{\leq}$ is a MBWST. ■

Thus, the algorithm runs in $O(m+n)$ time and outputs a MBWST. □