H8.

- The assignment is due at Gradescope on March 23 at 1:35pm. Late assignments will not be accepted. Submit early and often.
- You are permitted to study with friends and discuss the problems; however, *you must write up you own solutions, in your own words*. Do not submit anything you cannot explain. If you do collaborate with any of the other students on any problem, please do list all your collaborators in your submission for each problem.
- Finding solutions to homework problems on the web, or by asking students not enrolled in the class is strictly prohibited.
- We require that all homework submissions are prepared in Latex. If you need to draw any diagrams, however, you may draw them with your hand.

PROBLEM 1 *Borůvka*

We are going to develop a new algorithm for computing minimum spanning trees (MST) for graphs in which all edge weights are unique. Our new algorithm does not need a heap or auxiliary data structure.

(a) Argue that the lightest edge emanating from a vertex must be part of a minimum spanning tree.

Solution:

(b) The observation above is the main idea in the following algorithm for MST due to Borůvka. For each vertex, finds its minimum incident edge. As argued above, that edge belongs to the MST. The algorithm adds all those edges to the MST. Next, it uses DFS to find connected components formed by those MST edges. Each component is then merged into a single "super" vertex. When vertices are combined, all of the edges incident to those vertices are now incident to the corresponding "super" vertex. For example, in the left graph below, we add the edge (b, e) to the MST and contract nodes b and e into be. That creates two edges between be and c.

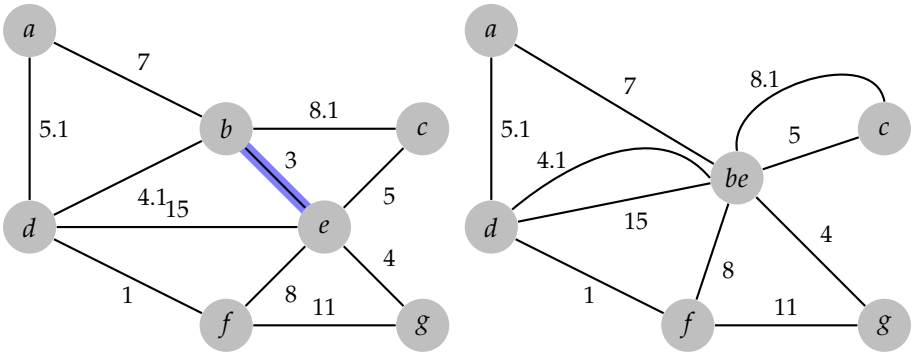

Figure 1: The graph G to the left, and an example of how nodes b and e can be contracted in G on the right.

1: **procedure** BORŮVKA($G = (V, E), w$)
2: $S \leftarrow \emptyset$, $V_0 = V$, $E_0 = E$, $i = 0$

```
3:      while |V_i| > 1 do
4:          for each v ∈ V_i do
5:              Find the lightest edge e_v = (v, w_v) emanating from v.
6:              Add e_v to the set S
7:          end for
8:          Run a DFS using only edges e_v to identify connected components.
9:          Merge all nodes in each connected component into one node.
10:         Rewrite edges using new nodes to produce updated graph G_{i+1} = (V_{i+1}, E_{i+1}).
11:         i ← i + 1
12:     end while
13:     return S
14: end procedure
```

Compute the result of running one iteration (lines 4–11) on graph G (list the edges e_v and the connected components).

Solution:

(c) What is the maximum number of vertices left after one iteration? Justify your answer.

Solution:

(d) If there are V_i vertices and E_i edges in the graph at iteration i, how much time does it take to execute the next iteration (lines 4–11)?

Solution:

(e) What is the overall running time of the algorithm? (Hint: determine the maximum number of iterations.)

Solution:

PROBLEM 2 *Feedback edge set*

Consider a undirected graph $G = (V, E)$ with positive edge weights $w : E \to \mathbb{R}$. We assume that G is connected.

(a) Give an algorithm for finding the maximum weight spanning tree of G. Hint: construct a new graph such that finding the maximum weight spanning tree in G is equivalent to finding the minimum weight spanning tree in the new graph. Your construction algorithm should run in $O(V + E)$ time.

Solution:

(b) A *feedback edge set* of an undirected graph G is a subset F of the edges such that every cycle in G contains at least one edge in F. In other words, removing every edge in F makes the graph G acyclic. Describe and analyze a fast algorithm to compute the minimum weight feedback edge set of G. Hint: prove that for any optimal solution F^*, $E \setminus F^*$ (the edges left after removing F^*) form a spanning tree in G. That is, you need to prove that $E \setminus F^*$ is acyclic and connects all vertices in G.

Solution: