Notes on Boruvka's Algorithm

All MST algorithms implement the invariant of the Cut Property. Kruskal's algorithm does it by always choosing, from among all shortest edges leaving a component of the tree X under construction, the one that is the shortest. Prim's algorithm always pick the shortest edge leaving the component that contains S, the (arbitrary) starting point.

Boruvka's algorithm for the MST goes all the way: It adds to X *all* shortest edges leaving *all* connected components of X. That is, at each stage it finds the connected components of (V, X), for each component it finds the shortest edge leaving it, and then it adds all these edges to X.

But there is a problem. If there are ties, then the algorithm can create cycles. (For example, imagine that all edges have length one, what will happen in the first step?) This is easy to fix: Add a tiny amount to all lengths so they are all different. Obviously, this does not change the MST.

Here is Boruvka's algorithm:

```
assumption:  all edge lengths are different
C is a set of connected components, initially {{1 },{2,},...,{|V|}}
X is the set of tree edges, initially empty
repeat
for each component c in C, find the shortest edge e leaving c, and add it to X
find the connected components of X
until |C| = 1 (there is only one component)
```

It takes $O(|E|)$ time to do each iteration (how do you find all shortest edges out of all components in linear time? hint: find the components first, then process all edges one by one, and look up the component of each endpoints, remembering for each component the shortest edge out of it that you have seen). Importantly, there are at most $\log |V|$ iterations (at each iteration the number of components $|C|$ is divided by two at least). So, this is an $O(|E| \log |V|$ algorithm, very much like the others.

It turns out that, of the three, this algorithm is the one that can be parallelized best, so we'll revisit it next week when we study parallel algorithms.