**Principles of Distributed Computing** 05/15, 2019

Exercise 11

*Lecturer: Mohsen Ghaffari*

## Cut Edges

**Exercise 1:** Given a graph $G = (V, E)$, an edge $e \in E$ is called a *cut edge* if its removal disconnects $G$. In this exercise, we devise an $O(D)$ round distributed algorithm (with $O(\log n)$-bit messages) that identifies all cut edges, where $D$ denotes the diameter of $G$.

Let $T$ be a BFS tree of $G$. For an edge $e \in T$ and an edge $e' \notin T$, we say $e'$ covers $e$ if and only if the unique cycle in $T \cup \{e'\}$ contains $e$. That is, in $T \cup \{e'\}$, there is a way to go from one endpoint of $e$ to its other endpoint, without passing through $e$. In fact, any nontree edge $e' = \{u, v\} \notin T$ covers all tree edges $e$ which are on the path from $u$ to $w$ (or from $v$ to $w$), where $w$ is the lowest common ancestor of $u$ and $v$.

(a) Argue that any cut edge in $G$ must be a tree edge $e \in T$ for which there is not nontree edge $e' \notin T$ that covers $e$.

(b) Devise an algorithm that in $O(D)$ rounds, makes each node $v$ know all of its at most $D$ ancestors.

(c) Use the above item to argue that in $O(D)$ rounds, we can make the endpoints $u, v$ of any nontree edge $e' = \{u, v\}$ know their lowest common ancestor $w$, at the same time for all nontree edges.

(d) Use the above items to complete an $O(D)$ round algorithm for identifying all cut edges.

**Solution:**

(a) First, note that any edge $e$ is a cut edge if and only if there is no cycle in $G$ that contains $e$. With this observation we can show the statement: A non-tree edge cannot be a cut edge, as otherwise the resulting tree $T$ would have to be disconnected. Also, any tree edge $e$ that is covered cannot be a cut edge, as there is a cycle containing $e$.

(b) In the first round, every node $v$ sends its ID to all its children. In each following round $i$, every node $v$ forwards the ID of its (unique) ancestor at distance $i$, i.e. the value it received in round $i - 1$ by its parent. This way, in $O(D)$ rounds, nodes learn the IDs of all their ancestors.

(c) First, note that a node can have at most $D$ ancestors, and that from (b) we can also assume that every node knows the ordering of its ancestors, starting from the root of $T$. For any non-tree edge $e' = \{u, v\}$, the endpoints $u$ and $v$ exchange all their ancestors in $O(D)$ rounds. Going through the ancestors in order starting from the root of $T$, the lowest common ancestor, or LCA, $w$ of $u$ and $v$ is the last node that is contained in both the ancestor lists of $u$ and $v$.

(d) For identifying the cut-edges we proceed as follows: Note that the previous items can easily be extended such that each for each $e = \{u, v\}$ we know both the LCA $w$ of $u$ and $v$, as well as its distance $h_w$ to the root. For any node $v'$, let $\ell(v')$ be the smallest distance $h_w$ amongst all $w = LCA(u, v')$, where $u$ is a non-tree neighbor of $v'$. If no such non-tree neighbor exists, let $\ell(v') = h_{v'}$. Further, let $min_v$ be the smallest value of $\ell(v')$ among

all descendants $v'$ of $v$. Now, every tree edge $e = \{u, v\}$, where $u$ is the parent of $v$, is a cut-edges iff $min_v > h_u$.

For performing this procedure in a distributed manner, all relevant values can be learned by convergecasting all the values $\ell(v)$ towards the root, always forwarding the smallest value.

## Orienting Long Trees

**Exercise 2:** Consider an arbitrary tree $T$ in a graph $G = (V, E)$. Suppose that $G$ has diameter $D$. However, $T$ can have arbitrarily large diameter, up to $n - 1$. Suppose that we are given a root node $r \in V$. We devise a randomized algorithm that in $O(D + \sqrt{n} \log n)$ rounds, orients all edges of $T$ toward the root $r$, with high probability. First, mark each edge of $T$ with probability $\frac{1}{\sqrt{n}}$.

(a) Prove that, with high probability, at most $O(\sqrt{n} \log n)$ edges are marked. If you cannot prove the statement with high probability, at least argue that the expected number of marked edges is at most $\sqrt{n}$.

(b) Prove that, with high probability, each of the connected components of $T$ remaining after the removal of marked edges has diameter at most $O(\sqrt{n} \log n)$.

(c) Devise an $O(\sqrt{n} \log n)$ round algorithm that identifies the connected components remaining after the removal of marked edges.

(d) Using the connected components identified above, devise an $O(D + \sqrt{n} \log n)$ round algorithm that orients all marked edges toward the root. Think about gathering marked edges and which components they connect to a central node.

(e) Assume that marked edges are oriented toward the root. Devise an algorithm that, in $O(D + \sqrt{n} \log n)$ extra rounds, orients the edges inside the components, all at the same time, toward the root.

**Solution:**

(a) We will argue that the expected number of marked edges is at most $\sqrt{n}$, the fact that at most $O(\sqrt{n} \log n)$ (or in fact even just $O(\sqrt{n})$) edges are marked with high probability, then directly follows from the application of Chernoff bound. For arguing about the expected value, for every $e \in T$, let $X_e$ be the indicator variable that is one iff $e$ is marked. Note that $\mathbb{E}[X_e] = \Pr[X_e = 1] = 1/\sqrt{n}$. The number of marked edges $X$ can then be expressed as $X = \sum_e X_e$, and by linearity of expectation we have $\mathbb{E}[X] = (n-1) \cdot 1/\sqrt{n} \leq \sqrt{n}$.

(b) Note that by definition of a tree $T$, for any two vertices $u, v$ there is exactly one path connecting $u$ and $v$ (in $T$). Let us consider $u, v$ at distance at least $10\sqrt{n} \log n$. For $u$ and $v$ to remain connected, none of the edges along their path in $T$ could have been marked. The probability for such an event to occur is at most

$$\left(1 - \frac{1}{\sqrt{n}}\right)^{10\sqrt{n} \log n} \leq e^{-\frac{10\sqrt{n} \log n}{\sqrt{n}}} = n^{-10}.$$

Union bounding over all (at most $\binom{n}{2}$ many) pairs $u, v$ at distance at least $10\sqrt{n} \log n$, we get that with probability at most $\binom{n}{2} n^{-10} \leq n^{-8}$ a component of diameter $10\sqrt{n} \log n$ remains.

(c) As components have diameter $O(\sqrt{n}\log n)$ with high probability, this can be achieved by just finding the node of smallest ID in each component, which will also act as the leader of this component for the following subtasks.

(d) First, we find a leader $c$ (e.g. the node with lowest ID) in $G$, and build a BFS tree $T'$ of $G$ rooted at $c$. Note that these things can be done in $O(D)$ time and that as opposed to $T$, which can have height $\ll D$, $T'$ has height at most $D$. Using $T'$, for each marked edge we send the edge, as well as the IDs of its adjacent components to the leader $c$. As at most $O(\sqrt{n}\log n)$ many edges are marked, this can be done in $O(D+\sqrt{n}\log n)$ rounds, by pipelining (with high probability). The leader $c$ now has the following view of the graph: It know all connected components formed by the non-marked edges, as well as all marked edges connecting them. Thus, it can orient the marked edges locally towards the component containing the root $r$ and distribute this information in $O(D+\sqrt{n}\log n)$ rounds.

(e) For each component, unless it contains the root $r$, there is exactly one incident marked edge that is oriented outwards of this component. Using BFS, we can now orient all other edges of each component towards the endpoint of this one marked edge, which is oriented towards the root. As components have diameter $O(\sqrt{n}\log n)$, this can be done in as many rounds.