Albert-Ludwigs-Universität, Inst. für Informatik
Prof. Dr. Fabian Kuhn

Distributed Systems, Summer Term 2020
Exercise Sheet 10

1 Gather everything in CONGEST

We are given a graph $G = (V, E)$ and we want each node to know the whole graph. We have seen during the lecture that such problem can be solved in $O(diam(G))$ rounds in the LOCAL model. Show that it is possible to solve this problem in $O(|E|)$ in the CONGEST model.

Sample Solution

Each node exchanges IDs with the neighbors: in this way, each node knows the set of its incident edges spending 1 round. Then, each node v:

- Stores a variable "$stopped(v)$" that at the beginning is set as $stopped(v) = False$.
- Keeps a structure (let us say a list), denoted with $L(v)$ where it puts its edges and all the edges that it receives (if any); after the first round $L(v) =$ list of incident edges.
- Keeps a structure of the sent edges $L'(v)$ initiated as empty.

We proceed by electing a leader and constructing a BFS tree rooted at the leader, which we can do in $O(diam(G))$ rounds. Then the algorithm proceeds as follows. Each node v (except the root):

- while $stopped(v) == False$,
  - sends to its parent msg($stopped(v)$, the first $O(1)$ incident edges of $L(v)$)
  - removes the sent edges from $L(v)$ and puts them in $L'(v)$
  - adds the edges received from children to $L(v)$ (ignoring duplicate edges)
  - If v is a leaf and $|L(v)| = 0$, it sets $stopped(v) = True$ and sends to its parent msg($stopped(v)$, $L(v)$).
  - If v is not a leaf and $|L(v)| = 0 \wedge \forall u$ child of v, v received $stopped(u) = True$, it sets $stopped(v) = True$ and sends to its parent msg($stopped(v)$, $L(v)$).
- When the root will receive from all its children their "*stopped*" flags set to True, it will know that it has received all the edges in the graph, hence the whole graph.

How much time does this procedure require? We need:

1. 1 round for the nodes to know their incident edges;
2. $O(diam(G))$ rounds for electing a leader and constructing the BFS tree;
3. $O(|E|) + O(diam(G))$ rounds for all edges to reach the leader/root.

The reason why point 3 requires $O(|E|)$ rounds is that, for each node v, $|L(v)| \leq |E|$, so each node can empty their list $L(v)$ in $|E|$ rounds and it will require $O(diam(G))$ rounds for the last $O(1)$ edges to reach the leader.
Once the leader has the all the edges, it can send them to the other nodes along the BFS tree using a similar procedure as the one described above: the root sends $O(1)$ edges per round to its children, and if it finishes it sets $stopped = True$ and sends this flag to the children; each node that is not the root or a leaf, at each round propagates all messages that it receives (since these are messages received over one edge, their size cannot be more than $O(\log n)$ bits) and if it receives $True$ from the parent at round r, at round $r+1$ it sends the last $O(1)$ edges and the flag set to $True$ to its children. For reasons similar as above, this procedure requires $O(|E|)$ rounds, which is the running time of the whole procedure.

2 APSP: Slow token

During the lecture we have seen two animations of the APSP algorithm:

- One where the token is moved slowly (every 2 rounds), and there are no waves that collide.
- One where the token is moved fast (every round), and many waves collide.

Prove that it is indeed true that, if we move the token every two rounds, each node at each round has to propagate at most one wave.

Sample Solution

We want to show that at each round, each node needs to propagate one wave: this ensures that there are no collisions of waves hence there is no congestion. We will show this by contradiction. Suppose that there exists a node v that at a certain round has to propagate at least 2 waves, W_1 and W_2. Let w_1 and w_2 be the nodes that started the waves W_1 and W_2 respectively. Recall that once a node v receives a wave W, it knows $d(v, w)$, that is the distance between node v and the node that started wave W (i.e., node w). Hence, after receiving W_1 and W_2, node v knows $d(v, w_1)$ and $d(v, w_2)$. Suppose $d(v, w_1) = x$ and $d(v, w_2) = y$. Notice that $x \neq y$, since otherwise it would mean that two nodes at the same distance from v, at the same time start a wave, but this is not possible since we have a unique token in the graph and only the node that holds the token can start a wave. So suppose w.l.o.g. that $x > y$. Also note that the distance in the BFS tree between nodes w_1 and w_2 is at most $(x - y)/2$ (since we move the token every two rounds). By triangle inequality we get that $x \leq y + (x - y)/2$, that is $x \leq y$, which is a contradiction.

3 Edge counting

Show that, in the CONGEST model, it is possible to count the number of edges of the graph in $O(diam(G))$ rounds.

Sample Solution

We start by electing a leader and constructing a BFS tree rooted at the leader. Then each leaf sends its degree to its parent. A non-leaf node waits to receive all messages from its children, sum all up (the received values plus its own degree), and sends the result to its parent. Once the root receives a messages from all children, it sums all up, let this sum be x. Since each node has counted each incident edge, it means that each edge is counted exactly twice. Hence the leader/root computes $x/2$ broadcasts it to the nodes in the tree. Overall we spend $O(diam(G))$ rounds.

4 Bipartite graph

Show that, in the CONGEST model, it is possible to detect if the graph is bipartite or not. In particular, all nodes have to output 1 if the graph is bipartite, and 0 otherwise. Show an algorithm that solves this problem in $O(diam(G))$ rounds.

Sample Solution

The idea is to use the fact that any bipartite graph is 2-colorable, and any graph that is 2-colorable is a bipartite graph. We start by electing a leader and constructing a BFS tree rooted at the leader. Now we 2-color the BFS tree in the following way: leader picks color 1 and informs its children. Then each other node, once it receives the color of the parent, they pick the color that is different from the parent's one, and if they have children they forward their color to the children. In $O(diam(G))$ we get a 2-colored BFS tree—notice that a 2-coloring of G implies a 2-coloring of the BFS tree, and 2-coloring of the BFS tree implies a 2-coloring of G if G is bipartite. So now what is left is to check whether the two coloring is actually a 2-coloring of the graph. For this, nodes spend 1 round and exchange their color with the neighbors. If a node that notice an inconsistency on the coloring, it will set a variable $temp - out = 0$, otherwise $temp - out = 1$. Now, as in the previous exercises, we will collect these "temporary outputs" from leaves towards the root, where each non-leaf node will compute the logical "and" between their temporary output and the ones of their children. In $O(diam(G))$ rounds the root will know whether the graph is bipartite or not and it will propagate this information to other nodes using for example the BFS tree. In total we spent $O(diam(G))$ rounds.